НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«На правах рукопису»
УДК 004.582

«До захисту допущено»
Науковий керівник кафедри
_____ І.А. Дичка
«___»_____2019 р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 121 Інженерія програмного забезпечення

на тему: «Спосіб та програмне забезпечення автоматизованого тегування текстових документів»

Виконав:
студент ІІ курсу, групи КП-81мп
Лисогор Дмитро Юрійович _____

Керівник:
Доцент кафедри ПЗКС, к.т.н., доцент,
Заболотня Т.М. _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н., доцент
Онай М.В. _____

Рецензент:
Доцент кафедри ММСА, к.т.н., доцент,
Дідківська М.В. _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2019 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою

Спеціальність (освітня програма) – 121 «Інженерія програмного забезпечення» ("Інженерія програмного забезпечення комп'ютерних та інформаційно-пошукових систем")

ЗАТВЕРДЖУЮ

Науковий керівник кафедри

_____ І.А. Дичка

«___»_____2018 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Лисогору Дмитру Юрійовичу

1. Тема дисертації «Спосіб та програмне забезпечення автоматизованого тегування текстових документів», науковий керівник дисертації Заболотня Тетяна Миколаївна, к.т.н., затверджені наказом по університету від «13» листопада 2019 р. № 3895-С.

2. Термін подання студентом дисертації «16» грудня 2019 р.

3. Об'єкт дослідження: процес використання тексту для аналізу та автоматизованого тегування даних.

4. Предмет дослідження: методи та підходи до класифікації та тегування текстів.

5. Перелік завдань, які потрібно розробити:
   - вивчення специфіки текстових документів;
   - формулювання вимог до розроблюваної моделі для автоматизованого тегування текстових документів;
   - оброблення вхідних даних та пошук додаткових параметрів в тексті для аналізу на основі геопозицій;
   - розроблення моделі для аналізу та автоматизованого тегування текстових документів;
   - порівняння розробленої моделі з існуючими, класичними;
   - визначення вимог до відповідного програмного забезпечення, яке реалізує запропоновану модель;
   - описати бізнес-модель, що дозволить представити на ринку повноцінний програмний продукт із використанням представлених у роботі напрацювань.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу:
   - загальна архітектура підходу;
   - схема організації крос-платформного тексту програми у розроблюваній системі;
   - діаграма класів каналу комунікації;
   - результати аналізу розробленого програмного забезпечення;
   - дерево проблем та рішень.

7. Орієнтовний перелік публікацій:
   - Тези доповіді “Особливості реалізації програмних моделей для задачі автоматизованого тегування опису вакансій”

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., к.т.н., доцент		

9. Дата видачі завдання «25» жовтня 2018 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Ґрунтовне ознайомлення з предметною галуззю	17.11.2018	
2.	Визначення структури магістерської дисертації; вивчення літератури, пошук додаткової літератури, патентний пошук	04.12.2018	
3.	Робота над першим розділом магістерської дисертації; проведення наукового дослідження	15.02.2019	
4.	Проведення наукового дослідження; робота над другим розділом магістерської дисертації; розроблення програмного забезпечення	05.04.2019	
5.	Проведення наукового дослідження; робота над статтею за результатами наукового дослідження	15.05.2019	
6.	Проведення наукового дослідження; робота над третім розділом магістерської дисертації	15.06.2019	
7.	Завершення роботи над основною частиною магістерської дисертації; підготовка ілюстративного матеріалу; підготовка матеріалів доповіді на конференції ПМК-2019	13.11.2019	
8.	Оформлення текстової і графічної частини магістерської дисертації	05.12.2019	

Студент Д.Ю. Лисогор

Науковий керівник дисертації Т.М. Заболотня

РЕФЕРАТ

Актуальність теми. Ринок пошуку текстових документів є великим і потужним механізмом, де зустрічаються інтереси компаній та шукачів ресурсів. Часто для того, аби знайти документ чи публікацію, що відповідає очікуванням людини, потрібно витратити багато часу. Найпершим підходом до спрощення вирішення задачі пошуку текстових документів є використання інформаційних технологій (зокрема баз даних, відповідних сайтів тощо), які набагато прискорюють процес пошуку потрібної інформації. Але разом з тим неправильне розуміння спільного та відмінного між текстовими документами, зв'язків між ними може призвести до витрати зайвих грошей, часу та ресурсів у процесі пошуку необхідної інформації. Таким чином, на разі актуальною є задача підвищення точності категоризації текстових документів для забезпечення користувачів більш релевантною інформацією в процесі пошуку документів чи публікацій.

Існує багато методів визначення категорій, до яких відноситься заданий текст. Однак більшість з них не можуть бути застосовані до текстових документів, оскільки тексти документів характеризуються наявністю великого набору прихованих категорій, які залежать від локації та часу публікації. Через це класичні моделі з оброблення текстових даних матимуть значні похибки у результатах класифікації, адже вони не враховують усіх особливостей таких текстових описів. З огляду на це, необхідним є врахування цих особливостей при розробленні моделей автоматизованого тегування текстових документів, адже пошук у попередньо тегованих даних є більш точним та швидким.

Об'єктом дослідження є процеси використаннятексту для аналізу та автоматизованого тегування даних.

Предметом дослідження є методи та способи до класифікації та тегування текстів.

Мета роботи: вдосконалення процесу пошуку категорій у тексті для отримання точних результатів пошуку категорій текстових документів шляхом розроблення та впровадження нового методу для автоматизованого виділення тегів.

Методи дослідження. В роботі використовуються методи математичного моделювання, методи оптимізації, методи класифікації текстових даних та методи машинного навчання.

Наукова новизна роботи полягає в наступному:

1. Розроблено спосіб автоматизованої категоризації текстових документів, що відрізняється від існуючих комбінацією існуючих класичних моделей класифікації текстових даних, які показують точніший результат виконання.

2. Запропоновано, ансамбль моделей для автоматизованої категоризації даних який відрізняється від відомих меншою обчислювальною складністю та точнішим результатом роботи з даними, які залежать від часу.

Практична цінність отриманих в роботі результатів полягає в тому, що запропоновані методи та засоби дають змогу отримати точніші результати під час пошуку та аналізу текстових описів. Розроблені методи, математичне і програмне забезпечення для автоматизованого тегування текстових описів, забезпечують комплексність вирішення задач пошуку у тексті, сприяють обґрунтованості знаходження результатів проаналізованих текстів.

Апробація роботи. Основні положення і результати роботи були представлені та обговорювались на XII науковій конференції магістрантів та аспірантів «Прикладна математика та комп’ютинг» ПМК-2019 (Київ, 13-15 листопада 2019 р.).

Структура та обсяг роботи. Магістерська дисертація складається з вступу, п’яти розділів, висновків та додатків.

У вступі надано загальну характеристику роботи, виконано оцінку сучасного стану проблеми, обґрунтовано актуальність напрямку досліджень, сформульовано мету і задачі досліджень наведено відомості про апробацію результатів і їх впровадження.

У першому розділі розглянуто основні методи класифікації текстових даних, які можуть бути використані для автоматизованої категоризації текстових описів; наведені теоретичні засади щодо побудови моделі машинного навчання та класифікаторів текстів.

У другому розділі проаналізовано критерії оптимізації існуючих методів класифікації текстових даних; досліджено методи оптимізації, які можуть бути використані для побудови ансамблю моделей та класифікаторів машинного навчання для текстової обробки; розглянуто математичні методи аналізу великих масивів даних з позицій системного підходу для виявлення категорій у тексті.

У третьому розділі сформульовано основні методичні підходи до розроблення ансамблю моделей і обраних класифікаторів (збір даних, обробка, підрахунок коефіцієнтів для кожного алгоритму, аналізатор рішень) для розв’язання задач автоматизованого аналізу текстових описів.

У четвертому розділі запропоновано алгоритм побудови ансамблю моделей; досліджено проблеми, пов’язані з аналізом і класифікацією текстових даних; запропоновано програмну реалізацію комбінацій моделей машинного навчання для обробки текстових даних.

У п’ятому розділі було сформовано та побудовано бізнес-модель кінцевого продукту, що описує ключові моменти в організації діяльності, пов’язаної з поширенням розробленої системи для автоматизованого тегування текстових документів.

У висновках проаналізовано отримані результати роботи.

Робота виконана на 90 аркушах, містить 3 додатки та посилання на список використаних літературних джерел з 26 найменувань. У роботі наведено 24 рисунки та 3 таблиці.

Ключові слова: метод оптимізації, модель машинного навчання, ансамбль моделей та класифікаторів текстових описів.

ABSTRACT

Actuality of theme. The text search market is a large and powerful mechanism for meeting the interests of companies and resource seekers. It often takes a long time to find a document or publication that meets human expectations. The first approach to simplifying the task of searching for text documents is to use information technologies (such as databases, relevant sites, etc.) that greatly speed up the search process. But at the same time, misunderstandings of what is common and what is different between these documents, the links between them can lead to waste of extra money, time and resources in the process of finding the necessary information. Thus, the task at hand is to improve the accuracy of categorizing text document data to provide users with more relevant information when searching for documents or publications. There are many methods for defining the categories to which a given text applies. However, most of them cannot be applied to the textual descriptions of documents, since these texts are characterized by the presence of a large set of hidden categories, which depend on the location and time of publication. Because of this, classic text-processing models will have significant errors in classification results because they do not take into account all the features of such text descriptions. With this in mind, it is necessary to take these features into account when designing automated text document tagging, because the search for pre-tagged data is more accurate and faster.

Object of research is the processing of textual data and the automatic classification of textual data.

Subject of research is methods and approaches to classifying and tagging texts.

Research objective: To improve the process of searching categories in text for accurate results by developing and implementing a new method for automated tag extraction.

Research methods. The methods of mathematical modeling, methods of optimization, methods of classification of text data and methods of machine learning are used in the work.

Scientific novelty of the work is as follows:

1. A method of automated categorization of text descriptions is developed, which is different from the existing combination of existing classic models, which show a more accurate result of execution.
2. An ensemble of models for automated categorization of data is proposed, which differs from the known ones with less computational complexity and more accurate result of working with time dependent data.

The practical value of the results obtained is that the proposed methods and tools make it possible to obtain more accurate results when searching and analyzing text descriptions. The developed methods, mathematical and software for automated tagging of text descriptions, provide complexity of solving problems of search in the text, contribute to the validity of finding the results of the analyzed texts.

Approbation. The main provisions and results of the work were presented and discussed at the XII Scientific Conference of Undergraduate and Graduate Students in Applied Mathematics and Computing PMK-2019 (Kyiv, November 13-15, 2019).

Structure and content of the thesis. The master's thesis consists of an introduction, five sections, conclusions and appendices.

The introduction gives a general description of the work, made an assessment of the current state of the problem, substantiated the relevance of the direction research, the purpose and tasks of the research are formulated, the scientific novelty of the obtained results and the practical value of the work are shown, information about the approbation of the results and their implementation is given.

The first section discusses the basic methods of classifying textual data that can be used to automate the categorization of textual descriptions; theoretical principles for the construction of machine learning model and text classifiers are given.

The second section analyzes the optimization criteria for existing textual data classification methods; optimization methods that can be used to construct an ensemble of machine learning models and classifiers for text processing are explored; mathematical methods of analysis of large arrays of data from the standpoint of systematic approach for identifying categories in the text are considered.

The third section sets out the basic methodological approaches to developing an ensemble of models and selected classifiers (data collection, processing, coefficient calculation for each algorithm, decision analyzer) to solve the tasks of automated textual description analysis.

The fourth section proposes an algorithm for constructing an ensemble of models; explored issues related to the analysis and classification of textual data; software implementation of combinations of machine learning models for word processing is proposed.

In the fifth section, a business model of the final product was created and built that describes the key points in organizing the activities related to the spread of the developed system for automated tagging of text documents.

The applications provide auxiliary functions for manipulating the internal state of plugins, examples of platform-specific plugins.

The work is made on 90 sheets, contains 3 appliccations and links to the list of used literary sources of 26 titles. The paper presents 24 figures and 3 tables.

Keywords: optimization method, machine learning model, model ensemble and text descriptors.

РЕФЕРАТ

Актуальность темы. Рынок поиска текстовых документов является большим и мощным механизмом, где встречаются интересы компаний и работодателей ресурсов. Часто для того, чтобы найти документ или публикацию, соответствует ожиданиям человека, нужно потратить много времени. Первым подходом к упрощению решения задачи поиска текстовых документов является использование информационных технологий (в частности баз данных, соответствующих сайтов и т.п.), гораздо ускоряют процесс поиска нужной. Но вместе с тем неправильное понимание общего и отличного между этими документами, связей между ними может привести к расходу лишних денег, времени и ресурсов в процессе поиска необходимой информации. Таким образом, на данный момент актуальной является задача повышения точности категоризации данных о текстовые документы для обеспечения пользователей более релевантной информацией в процессе поиска документов или публикаций.

Существует много методов определения категорий, к которым относится заданный текст. Однако большинство из них не могут быть применены к текстовых описаний документов, поскольку эти тексты характеризуются наличием большого набора скрытых категорий, которые зависят от локации и времени публикации. Поэтому классические модели по обработке текстовых данных будут иметь значительные погрешности в результатах классификации, ведь они не учитывают всех особенностей таких текстовых описаний. Учитывая это, необходимо учета этих особенностей при разработке моделей автоматизированного пометка текстовых документов, ведь поиск в предварительно тегированных данных является более точным и быстрым.

Объектом исследования являются процессы обработки текстовых данных и автоматизированная классификация текстовых данных.

Предметом исследования являются методы и подходы к классификации и пометка текстов.

Цель работы: совершенствование процесса поиска категорий в тексте для получения точных результатов путем разработки и внедрения нового метода для автоматизированного выделения тегов.

Методы исследования. В работе используются методы математического моделирования, методы оптимизации, методы классификации текстовых данных и методы машинного обучения.

Научная новизна работы заключается в следующем:

1. Разработан способ автоматизированной категоризации текстовых описаний, отличающийся от существующих комбинацией существующих классических моделей, которые показывают точный результат выполнения.
2. Предложено, ансамбль моделей для автоматизированной категоризации данных не отличается от известных меньшей вычислительной сложностью и точным результатом работы с данными, которые зависят от времени.

Практическая ценность полученных в работе результатов заключается в том, что предложенные методы и средства позволяют получить более точные результаты при поиске и анализа текстовых описаний. Разработанные методы, математическое и программное обеспечение для автоматизированного пометка текстовых описаний, обеспечивают комплексность решения задач поиска в тексте, способствуют обоснованности нахождения результатов проанализированных текстов.

Апробация работы. Основные положения и результаты работы были представлены и обсуждались на ХИИ научной конференции магистрантов и аспирантов «Прикладная математика и компьютинг» ПМК-2019 (Киев, 13-15 ноября 2019).

Структура и объем работы. Магистерская диссертация состоит из введения, пяти глав, заключения и приложений.

Во введении дано общая характеристика работы, выполнена оценка современного состояния проблемы, обоснована актуальность направления

исследований, сформулированы цели и задачи исследований, показано научную новизну полученных результатов и практическую ценность работы, приведены сведения об апробации результатов и их внедрение.

В первом разделе рассмотрены основные методы классификации текстовых данных, которые могут быть использованы для автоматизированной категоризации текстовых описаний; приведены теоретические основы по построению модели машинного обучения и классификаторов текстов.

Во втором разделе проанализированы критерии оптимизации существующих методов классификации текстовых данных; исследованы методы оптимизации, которые могут быть использованы для построения ансамбля моделей и классификаторов машинного обучения для текстовой обработки; рассмотрены математические методы анализа больших массивов данных с позиций системного подхода для выявления категорий в тексте.

В третьем разделе сформулированы основные методические подходы к разработке ансамбля моделей и избранных классификаторов (сбор данных, обработка, подсчет коэффициентов для каждого алгоритма, анализатор решений) для решения задач автоматизированного анализа текстовых описаний.

В четвертом разделе предложен алгоритм построения ансамбля моделей; исследованы проблемы, связанные с анализом и классификацией текстовых данных; предложено программную реализацию комбинаций моделей машинного обучения для текстовых данных.

В пятом разделе было сформировано и построено бизнес-модель конечного продукта, описывает ключевые моменты в организации

деятельности, связанной с распространением разработанной системы для автоматизированного пометка текстовых документов.

В выводах проанализированы полученные результаты работы.

Работа выполнена на 90 листах, содержит 3 приложения и ссылки на список использованных литературных источников из 26 наименований. В работе приведены 24 рисунки и 3 таблицы.

Ключевые слова: метод оптимизации, модель машинного обучения, ансамбль моделей и классификаторов текстовых описаний.

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

Наївний баєсів класифікатор – це ймовірнісний класифікатор, що використовує теорему Баєса для визначення ймовірності приналежності спостереження (елемента вибірки) до одного з класів при припущенні (наївному) незалежності змінних [1].

F-оцінка – критерій ефективності, який у статистичному аналізі двійкової класифікації є мірою точності визначення категорій, які присутні в тексті [2].

Метод *k*-найближчих сусідів – метричний алгоритм для автоматичної класифікації об'єктів. Основним принципом методу найближчих сусідів є те, що об'єкт присвоюється тому класу, який є найбільш поширеним серед сусідів даного елемента [3].

Дерево ухвалення рішень (також може називатися деревами класифікацій або регресійними деревами) – використовується в галузі статистики та аналізу даних для прогнозних моделей. Структура дерева містить такі елементи: «листя» і «гілки». На ребрах («гілках») дерева ухвалення рішення записані атрибути, від яких залежить цільова функція, в «листі» записані значення цільової функції, а в інших вузлах – атрибути, за якими розрізняються випадки. Щоб класифікувати новий випадок, треба спуститися по дереву до листка і видати відповідне значення. Подібні дерева рішень широко використовуються в інтелектуальному аналізі даних. Мета полягає в тому, щоб створити модель, яка прогнозує значення цільової змінної на основі декількох змінних на вході [4].

Фреймворк – це реальна або концептуальна структура, призначена слугувати опорою або керівництвом для побудови чогось, що розширює структуру на щось корисне.

ВСТУП

Ми живемо в епоху великих даних (від англ. «big data»), що постійно зростають в обсязі, а відтак вимагають застосування до них спеціальних методик для автоматизованої категоризації текстових даних. Разом з цим невпинно збільшується кількість користувачів, що хочуть отримати доступ до певної групи текстів (наприклад публікації на форумах, тексти описів вакансій, тощо), та, відповідно, зростає число запитів до програмних систем, які зберігають ці дані, відповідно це впливає на час виконання. Отже, цілком логічним є існування методів категоризації текстів, які будуть працювати у високонавантаженому режимі. Зважаючи на постійний характер зростання обсягів даних, кількості користувачів та запитів до пошукових систем, безперервний моніторинг рівня ефективності роботи останньої, а також визначення доцільних засобів та способів для віднесення певних текстових документів до відповідної категорії є важливою та актуальною задачею інженерів з розробки програмного забезпечення для автоматизованого тегування текстових описів.

На сьогоднішній день існує багато різноманітних алгоритмів для автоматизованої категоризації текстових документів. Однак, кожен з наявних засобів вимагає спеціальної підготовки розробників та/або адміністраторів бази даних. Вдосконалення процесу пошуку категорій та підвищення точності знайдених результатів для користувачів шляхом розроблення та впровадження нової моделі для автоматизованого виділення тегів текстових описів дозволить оптимізувати процес пошуку даних.

Існує багато алгоритмів для визначення категорій, до яких відноситься заданий текст. Проте більшість з цих алгоритмів не можуть бути застосовані для текстових описів документів, оскільки ці тексти характеризуються наявністю великого набору прихованих категорій, які залежать від часу публікації чи статті. Через це класичні алгоритми для оброблення текстових даних матимуть значні похибки у результатах класифікації, адже вони не

враховують усіх особливостей текстових документів. З огляду на це, необхідним є врахування цих особливостей при розробленні способів та програмного забезпечення автоматизованого тегування текстових документів, адже пошук у попередньо класифікованих даних є більш точним та швидким.

1. ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1. Огляд існуючих рішень для класифікації

Класифікація тексту (категоризація тексту або тегування тексту) – це завдання присвоєння набору попередньо визначених категорій довільному тексту [5]. Класифікатори текстових документів використовуються для впорядкування, структурування та категоризації майже будь-яких текстових описів. Наприклад, нові статті можуть бути організовані за темами, розмови в чаті можна організувати за мовою, згадки про бренд можна організувати за настроями, тощо.

Як приклад, наступний текст нижче:

“The user interface is quite straightforward and easy to use.”

Якщо використати класифікатор для вхідного тексу, можна проаналізувати його вміст, а потім і автоматично призначити відповідні теги, такі як “UI” та “Easy To Use”, які представляють цей текст на рис. 1.1.

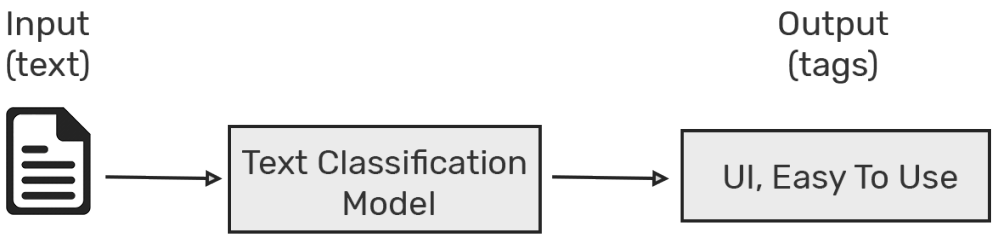


Рис. 1.1. Приклад роботи текстового класифікатора

Класифікацію тексту можна здійснити двома різними способами: вручну та автоматично. У першому способі, відповідна особа інтерпретує текст так відповідно виконує його класифікацію. Цей метод зазвичай може забезпечити якісні результати, але це трудомісткий і дорогий шлях. Для автоматизованої класифікації текстових документів використовується машинне навчання, природну обробку мови для швидкого та економічно вигідного результату.

Існує декілька способів до автоматизованої класифікації текстових документів, які можна згрупувати у три різні типи:

- Системи на основі правил.
- Системи на основі машинного навчання.
- Гібридні системи.

Розглянемо описану вище класифікацію систем більш детально.

1.1.1. *Системи на основі правил*

Підходи, засновані на наборі лінгвістичних правил створених анотатором, класифікують текст на організовані групи. З цих правил система використовує семантично відповідні елементи тексту для знаходження відповідних категорій завдяки індентифікації. Кожне правило визначається шаблоном та передбачуваною категорією.

Наприклад, необхідно класифікувати статті новин за двома категоріями, а саме: «Спорт» та «Політика». По-перше, для цього необхідно визначити два списка слів, які характеризують кожну окрему категорію новин (наприклад, слова, пов’язані зі спортом, такими як футбол, баскетбол, Леброн Джеймс тощо), а також слова, пов’язані з політикою, такі як Дональд Трамп, Хілларі Клінтон, тощо). Далі, коли необхідно класифікувати новий вхідний текст, необхідно провести підрахунок слів, які належать до окремих категорій, відповідно. Якщо кількість співпадінь зі словами словника, пов’язаних зі спортом, більше, ніж кількість слів, пов’язаних з політикою, текст відноситься до спортивної категорії.

Ця ж система з набором визначених правил класифікує заголовок «Коли перша гра Леброна Джеймса з Лейкерсами?» як спортивний, оскільки було знайдено слова, які відносятьс до цієї категорії.

Системи, засновані на правилах, прості та зрозумілі у використанні, але такий підхід має свої недоліки. Для початку ці системи потребують глибокого знання області. Їх розроблення також займає багато часу, оскільки генерування правил для складної системи може бути досить

важким і зазвичай вимагає багато часу для тестування та аналізу. Ці системи, також важко підтримувати і вони не можуть маштабуватися, враховуючи, що нові правила можуть вплинути на класифікацію попередніх категорій.

1.1.2. *Системи на основі машинного навчання*

Системи машинного навчання для класифікації текстових документів виконують класифікацію на основі попередніх спостережень. Використовуючи приклади даних які були попередньо розмічені, як навчальні дані, алгоритми машинного навчання вивчають різні асоціації між фрагментами тексту і що певний вихід (тобто теги) очікуються для конкретного вводу (тобто тексту).

Першим кроком до навчання класифікатора є вилучення особливостей, знайдених у текстових документах: використовується метод перетворення кожного тексту у числове подання у вигляді вектора. Один з найбільш часто використовуваних підходів – мішок слів, де вектор представляє частоту слова у заздалегідь визначеному словнику слів.

Наприклад, якщо визначити словник, щоб він містив такі слова:

- this;
- is;
- the;
- not;
- awesome;
- bad;
- basketball.

І необхідно векторизувати текст: “This is awesome”. Результатом буде векторне подання тексту, представлене в такому вигляді: (1, 1, 0, 0, 1, 0, 0).

Надалі алгоритм машинного навчання використовує навчальні дані, що складаються з набору визначених функцій (векторів для кожного окремого тексту) та тегів (наприклад, спорт, політика) для створення моделі класифікації. Алгоритм роботи даного способу зображено на рис. 1.2.

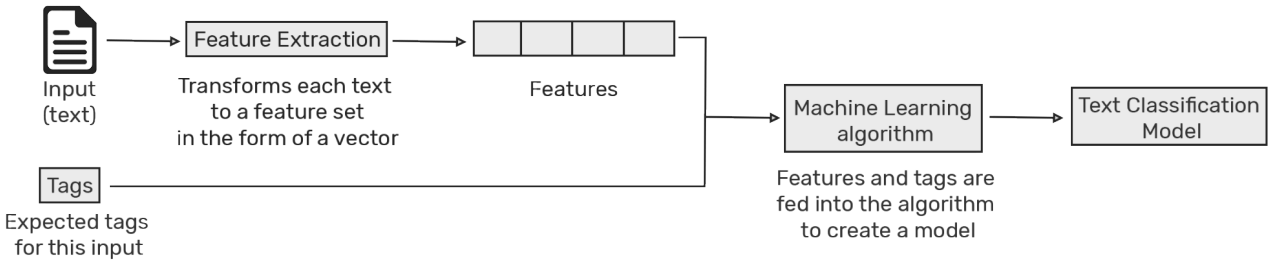


Рис. 1.2. Алгоритм роботи класифікатора машинного навчання

Після того, як класифікатор пройде навчання з достатньою кількістю зразків для тренування, модель машинного навчання буде здатна робити точні прогнози. Цей же екстрактор функцій використовується для перетворення невідомого тексту в набори векторів, які будуть подаватися у моделі для класифікації, щоб отримати прогнозовані категорії, наприклад, спорт, політика. Алгоритм роботи класифікатора після навчання на рис. 1.3.

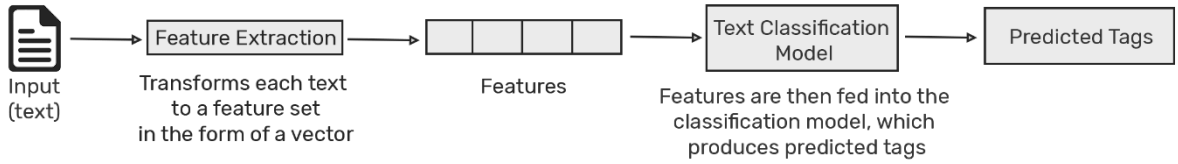


Рис. 1.3. Алгоритм роботи натренованого класифікатора тексту машинного навчання

Класифікація тексту з використанням моделей машинного навчання набагато точніша, ніж створені людиною набори правил, особливо у випадках складних завдань класифікації. Також такі класифікатори простіші у обслуговуванні, і завжди можна помітити нові категорії для

вивчення нових завдань. Тому розглянемо класичні алгоритми класифікації тексту – деякі з найпопулярніших алгоритмів машинного навчання для створення моделей класифікації тексту включають: сімейство алгоритмів Байєса, підтримуючі векторні машини та глибинне навчання.

- Naive Bayes – відноситься до сімейства алгоритмів, які можна використовувати для класифікації тексту. До цього сімейства відноситься баєсовий наївний алгоритм. Головна перевага, данного алгоритму – точність виконання алгоритму для маленьких об’ємів даних та вибірок. Алгоритм базується на теоремі Баєса, яка дозволяє отримати обчисленні ймовірності виникнення двох подій для кожно окремої взятої події. Для кожного вектора, який відображає текст, міститься інформація про ймовірність появи слів для тексту певної категорії для того щоб текст належав до цієї категорії.
- Support Vector Machines (SVM) – це лише один із багатьох алгоритмів, з яких можна обрати, коли необхідно виконати класифікацію тексту. Як і вище згаданий алгоритм, SVM не потребує великої кількості даних для навчання і отримання точних результатів. Хоч SVM потребує більшої кількості обчислюваних даних, проте в такому випадку він досягає точніших результатів. Такоє SVM будує два підпростори, які належать до певної групи чи ні відповідно. Дані вектори представляють зображення та теги навчальних текстів чи груп на якому позначили тексти.

- Deep Learning – це набір алгоритмів і прийомів, побудованих на принципі роботи людського мозку. Завдяки малій потребі в інженерних функціях, даний алгоритм набув популярності у даний момент часу. Даний алгоритм містить дві можливі архітектури, – це нейронні мережі конволюційні (CNN) та нейронні мережі періодичні (RNN). Хоча ці алгоритми потребують великої кількості розмічених даних, набагато більше ніж класичні алгоритми (близько мільйона розмічених прикладів). Проте, класичні алгоритми навчання моделі, такі як SVM та NB, мають деякий поріг, після якого додавання до них навчальних даних, не збільшується точність. Але такі класифікатори продовжують навчатися краще тоді, коли віддаєте їм більше даних, рис. 1.4.

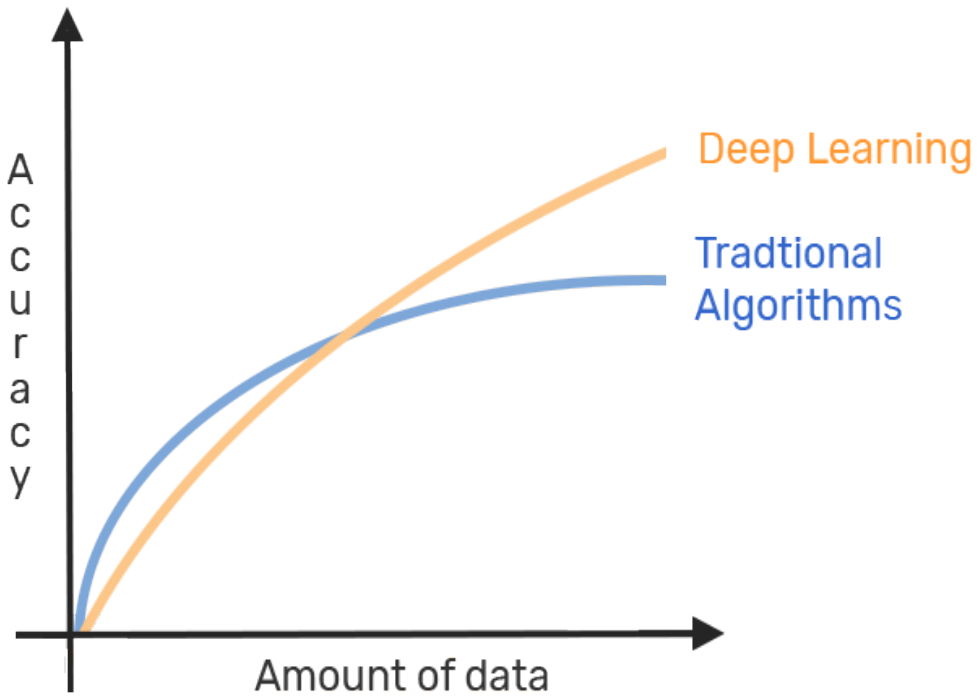


Рис. 1.4. Порівняльний графік точності алгоритмів машинного навчання

1.1.3. *Гібридні системи*

Гібридні системи поєднують базові навчальні класифікатори, та системи, побудовані на основі правил, які використовуються для покращення результатів навчання. Такі системи легко налаштовуються, шляхом додавання правил для тих проблемних тегів, які не були коректно протестовані базовими класифікаторами.

1.1.4. *Способи та методи оцінки рельзультатів роботи алгоритмів машинного навчання*

Перехресне підтвердження – є стандартним методом оцінки продуктивності роботи класифікаторів текстових даних. Він полягає у поділі навчального набору випадковим чином на набори примірників однакової довжини. Для кожного, окремого текстового набору, класифікатор навчається з іншими зразками (приблизно 70% прикладів). Далі класифікатори текстових документів прогнозують відповідні набори, а результати визначаються правильними чи ні, на основі розміток, які робить людина. Це допомагає визначити, коли результат був правильним (справжні позитивні і справжні негативні), а коли був хибним (помилкові позитиви, помилкові негативи).

Завдяки цим результатам можна визначити показники, які дозволяють визначити, наскільки добре працює класифікатор:

- Accuracy – відсоток текстів, які були передбачені з правильним тегом.
- Precision – відсоток прикладів, який класифікатор визначив із загального набору прикладів, які були предебачені для даного тегу.
- Recall – відсоток прикладів, які визначив класифікатор для відповідного тегу, які повинен був визначити серед загального набору даних.
- F1 score – середнє гармонійне значення точності та відхилення.

1.2. Огляд існуючих рішень для автоматизованої категоризації текстів

Розглянемо існуючі рішення для автоматизованого тегування текстових описів. Першим етапом для класифікації тексту – це підготовка тексту. Підготовка даних є важливим питанням класифікації. Цей процес включає наступні дії, такі як вибір, попередня обробка та трансформація.

- Вибір – цей етап вибору підмножини всіх доступних даних, з якими алгоритм буде працювати. Завжди є сильне прагнення включити всі доступні дані, щоб було «чим більше, тим краще», але необхідно врахувати, які саме дані знадобляться для вирішення проблеми. Необхідно отримати кілька припущень щодо потрібних даних та обережно їх використати, щоб записати ці припущення з метою, що користувач зможе перевірити їх пізніше, якщо це необхідно. Вибір даних визначався на основі наявних даних, відсутніх даних та даних, які необхідно видалити.
- Попередня обробка – попередня обробка даних може сортувати вибрані дані шляхом форматування, очищення, інтеграції та вибірки.
- Очищення даних – етап в якому заповнюються відсутні дані та відбувається переналаштування недійсних даних. Ідентифікуються залишки, наявні у даних, і з цих даних – невідповідності видаляються з джерела.
- Інтеграція даних – дані з різних джерел об'єднуються в єдину базу даних.
- Перетворення даних – вихідні дані перетворені у загальний формат, який використовується для обробки.
- Зменшення даних – це шлях до видалення небажаних параметрів із даних. Отже, обсяг даних буде меншим, не впливаючи на якість інформації.

- Дискретизація даних – частина процесу скорочення даних. Цей процес замінить числові атрибути номінальними атрибутами.
- Перетворення – трансформовані дані готові до виконання операцій над ними. Зазвичай існує три основні перетворення даних:
    - декомпозиція атрибутів;
    - агрегування атрибутів;
    - масштабування.

Далі розглянемо контрольовані алгоритми навчання для класифікації текстових описів.

1.2.1. *Дерева рішень*

Дерева рішень – це домінуюча класифікація алгоритмів, які значною мірою використовуються при обробці даних. Класифікація описується як блок-схема, схожа на структуру дерева, яка включає кореневий вузол, у нелистові вузли та листкові вузли. Кожен нелистовий вузол описує атрибут тесту, кожна гілка описує результат цього тесту, а кожен вузол містить мітку класу.

Фундаментальна ідея дерева рішень полягає в розділенні даних рекурсивно на підмножини з кінцевою метою, якою була кожна підмножина, яка містить приблизно однорідні стани цільової змінної. Вибір критерію розщеплення тобто міра вибору атрибутів повинна вибирати таку, щоб вона найкраще розділяла даний набір даних. Деякі з відомими алгоритмами дерева рішень є ID3, C4.5 та CART, які використовують інформаційний приріст, посилення впливу коефіцієнту співвідношення та індекс Джині відповідно до вимірювання вибору атрибутів. C5 – це також дерево рішень на основі алгоритму, який був розширеною версією C4.5. Коли дерево рішень побудоване, алгоритм використовується для класифікації іншого нового примірника, виконуючи операцію проходження від кореневого вузла до вузла кінцевого,

застосовуючи критерії тестування до кожного не листкового вузла. Клас листового вузла – клас для кожного екземпляра.

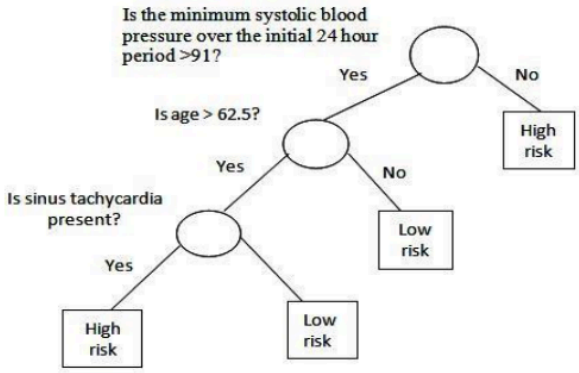


Рис. 1.5. Приклад дерева рішень

Переваги:

- Дерева рішень легко зрозуміти.
- Непрофесійні розробники можуть легко зрозуміти дерева рішень з розумною кількістю листя.
- Для кращого розуміння для користувачів дерева рішень перетворюються на правила.
- Дерева рішень можуть містити як числові, так і номінальні атрибути.

Недоліки:

- Більшість алгоритмів, таких як ID3 та C4.5, потребують наявності цільового атрибута – одиничні дискретні значення.
- Дерева рішень використовують метод поділяй і володій, який має схильність до досягнення успіху, якщо існує невелика кількість надзвичайно значущих атрибутів, за винятком меншого випадку, якщо доступні численні складні взаємодії.

1.2.2. *Класифікація на основі правил*

Класифікація, заснована на правилах, є однією з вивчених моделей, яка представляється як набір правил (if – then).

Правила складаються з двох частин, а саме: правило, що являє собою частину If, і правило, що є наслідком – що представляє іншу частину. Правила у формі: “if condition then conclusion”

Умова, згадана у вищенаведеному правилі, описувала атрибут тестів і містить передбачення класу.

Переваги:

- Як виняток, представляються як дерева рішень.
- Легко зрозуміти роботу системи.
- Швидка розробка системи.
- Можна швидко класифікувати нові екземпляри.
- Продуктивність схожа на дерева рішень.

Недоліки:

- Класифікатор створює правила з дуже малим покриттям.
- Правила можуть бути не вичерпними.

1.2.3. *Байєсівський класифікатор*

Байєсівські класифікатори відомі як статистичні класифікатори. Вони використовуються для прогнозування класу ймовірності членства, наприклад, ймовірність даного кортежу, який належить певному класу. Нехай X – набір даних, який називається доказом, а H – деяка гіпотеза.

Теорема Баєса є корисною для обчислення за задньою ймовірністю P (H | X) від P (H), P (X | H) і P (X). Існують такі класифікатори для вирішення задачі автоматизованого тегування текстових описів:

- Наївний байєсівський класифікатор – передбачає атрибути значень, які можуть бути умовно незалежними від іншого, а вони не відповідають цим існуючим атрибутам залежності між атрибутами. Це найточніший класифікатор, коли припущення

вірне. Припустимо, що є m класів, які представлені у вигляді C1, C2 ... Cm. Прогнозуючи, чи X належить до класу, який має найвищу попередню ймовірність, яку обумовлював X класифікатор. Класифікатор передбачить кортеж X, чи належить він до певного класу Ci у певному випадку.

- Байєсівська мережа передбачень (BBN) – класифікатор припускає, що атрибути не залежать один від одного, але в реальних сценаріях атрибути можуть співвідноситися, наприклад з медичною сферою, де співвідносяться симптоми пацієнта та стан здоров'я. Насправді залежності існують, і тому для моделювання залежностей між атрибутами за допомогою спільних залежностей використовуються байєсівські мережі передбачень для умовних розподілів ймовірностей. Класифікатор складається з:
    - залежностей серед атрибутів, які будуть представлені з використанням DAG (спрямований ациклічний граф)
    - умовна ймовірність – таблиця (CPT), яка пов'язана з кожним атрибутом. Вузли представляють атрибути та дуги, які представляють залежності. Припустимо, X – набір даних, що має *x1*…*xn* з деякими атрибутами *Y1*…*Yn*. Будь-які вузли, які є присутні в Байєсовій мережі та були умовно незалежними від своїх не нащадків, якщо їх батьки були відомі.

Переваги:

- Висока швидкість.
- Обробляє як числові, так і дискретні атрибути, що оцінюються.
- Легко обчислити.

Недоліки:

- Оптимальний класифікатор Байєса був обчислювально нерозбірливий.
- Наївні припущення Байєса зазвичай порушуються.

- Не дає точних результатів для деяких випадків, коли серед них існує залежність змінних.

1.2.4. *Логістична регресія*

Логістичні регресії – один із статистичних методів, що використовуються для аналізу наборів даних, які мають принаймні одну незалежну змінну для визначення результату. За допомогою логістичної функції ми можемо вимірювати взаємозв'язок між однією або кількома незалежними змінними та категоріально залежними змінними через оцінку їх ймовірностей. Результат є двозначним, що носить категоріальний характер. Крім того, він використовується для прогнозування ймовірності виникнення чи не виникнення події.

Переваги:

- Цей метод може уникнути значних відхилень при тренуванні.
- Можна здійснити вибір функції.
- Вихід можна трактувати як вірогідність.
- Стійкий до шуму в даних.

Недоліки:

- Для досягнення стабільних результатів потрібно більше даних.
- Визначення правильних незалежних змінних.
- Виникнення значних витрат ресурсів.

1.2.5. *K-Найближчих сусідів*

K-найближчих сусідів – класифікатор представлятиме кожен кортеж як один вузол даних у d-вимірному просторі, де d містить кількість атрибутів. Таким чином зберігається кожен кортеж та запам'ятовується під час етапу навчання. Коли кортеж новий, клас якого невідомий, KNN класифікатор порівнює свою близькість з *k*-найближчими навчальними кортежами і призначає клас *k*-найближчим сусідом з більшістю співпадінь. Деякі з широко застосовуваних заходів щодо пошуку найближчих сусідів

включають евклідову відстань Манхеттена, коефіцієнт простого узгодження, подібність та коефіцієнт кореляції. KNN має широкий спектр застосувань, що включає кластерний аналіз, аналіз зображень, розпізнавання шаблонів, прогнозування та економічне прогнозування.

Переваги:

- Швидке навчання.
- Наявність складних цільових функцій.
- Відсутність втрати інформації.
- Легкий у виконанні.

Недоліки:

- Повільно обробляє запит.
- Потрібно великий простір для запам'ятовування всіх примірників.
- Чутливий до шумів в даних.
- Повільне тестування.

1.2.6. *Штучні нейронні мережі*

Штучна нейронна мережа (ANN) – це обчислювальна модель, що залежить від біологічних нейронних систем. Вона включає взаємопов’язані елементи обробки, відомі як нейрони або вузли, які співпрацюють для спільної роботи для створення вихідної функції. Це адаптивна основа, яка змінює структуру, залежну від внутрішньої чи зовнішньої інформації, яка протікає через мережу та через навчання. ANN може бути простою моделлю перцептрона або більш складною багатошаровою моделлю перцептрона. Ця модель містить набір вихідних вузлів, де кожен вузол введення з'єднаний з вихідним вузлом через ваги. Модель тренується, регулюючи їх вагу, очікуючи, що вони відповідатимуть вхідно-вихідному співвідношенню даних. Багатошаровий ANN складається з декількох проміжних прихованих шарів, що з'єднують вхідний, вихідний шари і можуть бути використані для моделювання складних взаємозв'язків між вхідними, вихідними змінними. ANN успішно застосовується в галузі клінічної медицини в класифікації та

розпізнаванні образів. Нейронні мережі зі зворотним розповсюдженням та виділенням правил асоціації, що використовується для класифікації пухлин у мамограмах.

Переваги:

- Адаптивне навчання.
- Самоорганізація.
- Швидке прогнозування.
- Легко визначає складні стосунки.
- Може обробляти шумні дані.

Недоліки:

- Усі входи повинні бути перетворені в числові.
- Навчання дає місцевий оптимум.
- Виникнення значних витрат.
- Складно тлумачити.
- Час обробки значних нейронних мереж великий.
- Не можна ініціалізувати за допомогою попередніх знань.

1.2.7. *Підтримка векторної машини*

Вектори підтримки (SVM) були одним з контрольованих алгоритмів навчання, які є найбільш домінуючими алгоритмами класифікації в положеннях аналізу даних та точності прогнозування, які можуть бути використані як для класифікаційного, так і для регресійного аналізу. Він використовується для класифікації як лінійних, так і нелінійних даних.

Спочатку він був розроблений для двокласних задач, але пізніше він також використовувався для багатокласних проблем. Основним принципом SVM є – пошук оптимальної гіпер площини серед максимальної відстані до найближчої точки двох класів. Набір кортежів, найближчих до оптимальної гіпер площини, називався вектором опори. SVM використовує ці вектори підтримки для пошуку оптимальної гіпер площини. Пошук найкращої гіпер площини забезпечує лінійний класифікатор, тоді як для класифікації

нелінійних даних вихідні дані тренувань були перетворені у вищий розмір, використовуючи нелінійні функції ядра, такі як поліноміальний, радіальний, гауссовий, сигмоїдний і т.д. SVM застосовуються для числового прогнозування та класифікації. Вони мають широку область застосування, яка включає розпізнавання образів, медицину, біоінформатику, розпізнавання об'єктів та прогнозування. SVM використовується для кластеризації даних мікромасив та вилучення асоційованих генів для класифікації пов'язаних з браком документів.

Переваги:

- Детермінований алгоритм.
- Використовує максимальну граничну гіпер площину для класифікації лінійно відокремлюваних даних.
- Може вивчити дуже складні функції за допомогою ядер.

Недоліки:

- Важко тренується.
- Обчислювально дорогий алгоритм
- Навчальний процес може зайняти багато часу.
- Може вирішити задачу бінарного класу.

1.3. Висновки до розділу 1

У даному розділі розглянуто опис алгоритмів класифікації, таких як дерево рішень, класифікація на основі правил, байєсівський класифікатор, логістична регресія, KNN, ANN та SVM. Вирішено використовувати комбінацію вище згаданих методів для отримання необхідної точності для будь-якого набору текстових даних. Обговорено різні заходи щодо ефективності класифікації даних разом з різними інструментами для виконання алгоритмів класифікації на епоху великих даних. Перераховано переваги та обмеження різних алгоритмів класифікації, щоб забезпечити

можливість для подальших досліджень. Теоретичне підґрунтя, розглянуте у даному розділі, використовується у подальших розділах.

2. КОМБІНОВАНИЙ СПОСІБ АВТОМАТИЗОВАНОГО ТЕГУВАННЯ ТЕКСТОВИХ ДОКУМЕНТІВ

2.1. Огляд алгоритмів машинного навчання для реалізації

Проблеми з класифікацією – це не що інше, коли незалежні змінні є суцільними за своєю природою, а залежні змінні мають категоричну форму. Розглянемо декілька алгоритмів машинного навчання, які буде використано для вирішення задачі класифікації текстових документів.

2.1.1. *Логістична регресія*

Логістична регресія – один з найпоширеніших і корисних алгоритмів класифікації в машинному навчанні.

Логістична регресія – це тип алгоритму класифікації. Потрібно знати різницю між регресійними та класифікаційними завданнями, а також алгоритми, які слід використовувати в кожному. Завдання класифікації та регресії – це обидва типи контрольованого навчання, але вихідні змінні обох завдань різні. У задачі регресії вихідна змінна – це числове значення, яке існує в безперервному масштабі, або, якщо сказати іншим способом, то вихід регресійної задачі є цілим чи величиною з плаваючою точкою. На відміну від цього, у класифікаційній задачі результати алгоритму підпадають під одну з попередньо обраних категорій. Класифікаційна модель намагається передбачити вихідне значення, якщо дано кілька вхідних змінних, розмістивши приклад у правильній категорії. Розглянемо приклад класифікаційних завдань та регресійних завдань, щоб переконатися в різниці. Нехай, існує набір даних, повний докладно про різні будинки, і необхідно передбачити ціну, за яку продаватиметься будинок. У регресійному завданні модель бере такі особливості, як кількість кімнат, площа землі, вік будинку тощо, і намагається передбачити числове значення, наприклад, $95, 825. Математично для {x1, x2,…, xn} будучи ознаками, а y – ціною, припустимо функцію гіпотези 0 такою, що $y = 0$ (x1,

x2,…, xn). Регресійна модель спробує намалювати конкретний приклад, який найкраще підходить до набору даних. У класифікаційному завданні результати підпадають під одну з кількох різних категорій. Наприклад, було п'ять різних категорій за ціною:

- Набагато нижча за очікувану ціну.
- Приблизно очікувана ціна.
- Вище за очікувану ціну.
- Набагато вище очікуваної ціни.

Алгоритм класифікації позначить приклад однією з обраних категорій.

Логістична регресія – це алгоритм класифікації, який використовується, коли значення цільової змінної має категоріальний характер. Логістична регресія найчастіше використовується, коли дані мають двійковий вихід, тому коли вони належать до одного чи іншого класу, або це 0 або 1. Необхідно зауважити, що завдання класифікації мають дискретні категорії, на відміну від завдань регресії. Тут, ідеєю використання регресійної моделі для вирішення класифікаційної задачі, раціонально поставити питання про те, чи можна зробити функцію гіпотези такою, щоб вона підходила до двійкового набору даних. Для спрощення використовуємо лише проблеми бінарної класифікації із набором даних, зображених на рис. 2.1.

За допомогою логістичної функції ми можемо вимірювати взаємозв'язок між однією або кількома незалежними змінними та категоріально залежними змінними через оцінку їх ймовірностей. Результат є двозначним, що носить категоріальний характер. Крім того, він використовується для прогнозування ймовірності виникнення чи не виникнення події.

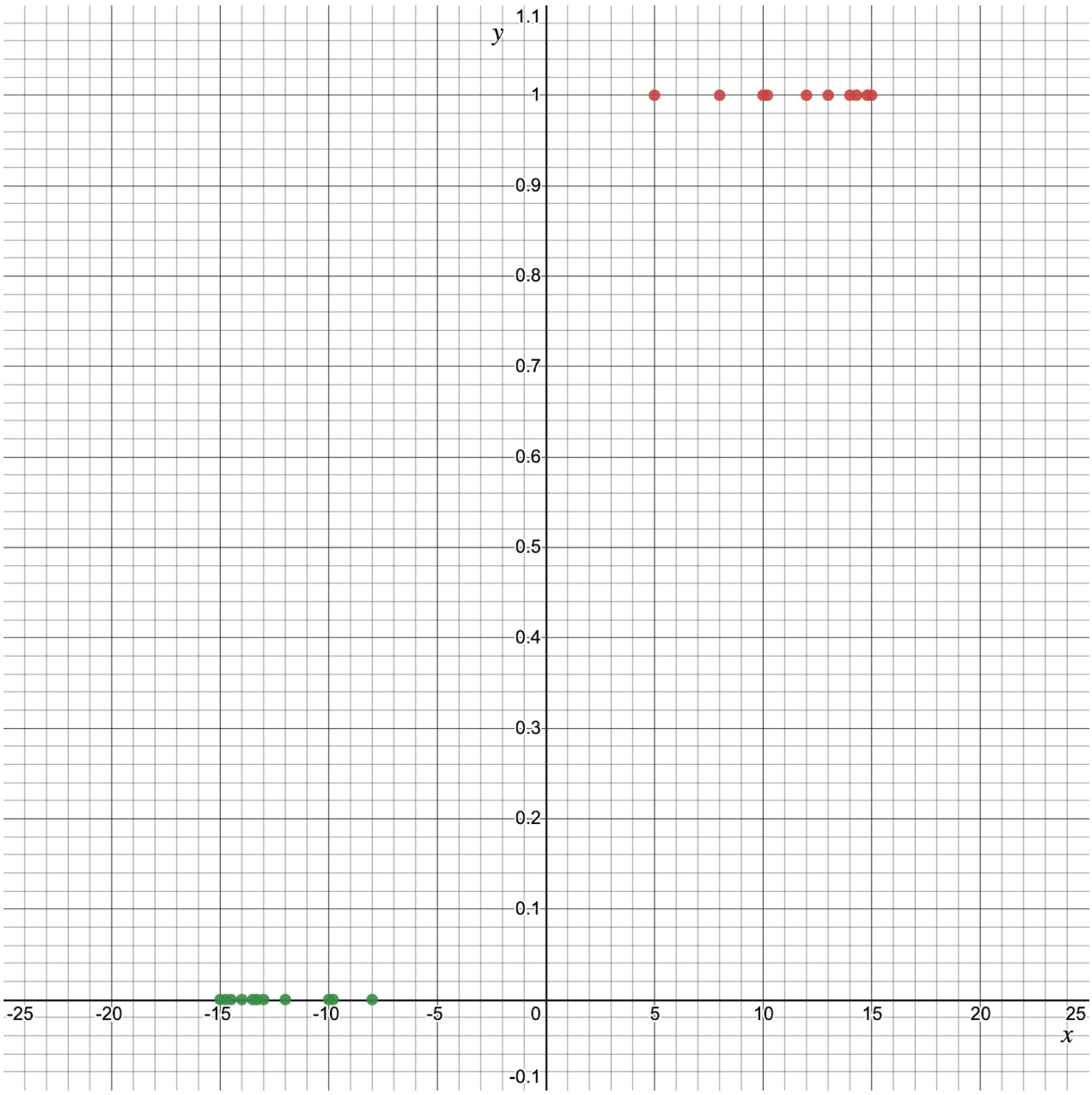


Рис. 2.1. Набір даних бінарної класифікації

Відповідь полягає в тому, що доведеться використовувати тип функції, відмінний від лінійних функцій, який називається логістичною або сигмоїдною функцією. Хоч алгоритм називається «Логістична регресія», це насправді алгоритм класифікації, а не алгоритм регресії. Сигмоїдна функція або логістична функція – це функція, яка нагадує криву у формі «S», якщо вона зображена на графіку. Вона приймає значення між 0 і 1 і прикріплює їх до краю вгорі і внизу, позначаючи їх як 0 або 1. На рис. 2.2 сигмоїдна функція представляє даний набір даних.

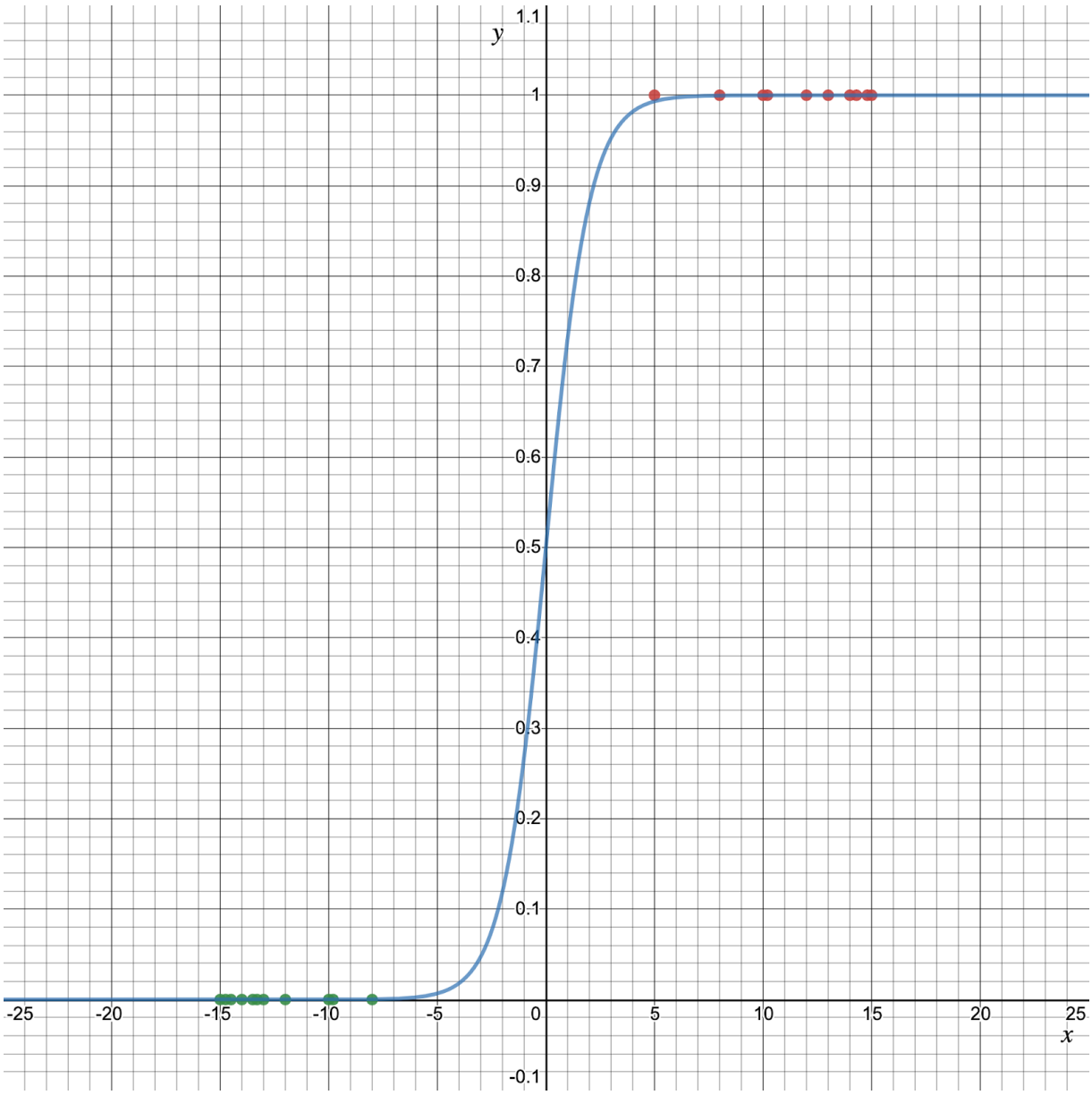


Рис. 2.2. Графічне зображення сигмоїдної функції

Це дає значення *y*, що надзвичайно близьке до 0, якщо *x* – велике від'ємне значення і близьке до 1, або якщо *x* – велике додатне значення. Після введення вхідного значення – від 0 до 1 вхід може бути виконаний за допомогою типової лінійної функції, але тепер вхідні дані можна ставити в різні категорії.

Логістичну регресію слід використовувати лише тоді, коли цільові змінні потрапляють у дискретні категорії, та якщо існує діапазон безперервних значень, цільове значення може існувати, то логістична

регресія не повинна використовуватися. Приклади ситуацій, в яких можна використовувати логістичну регресію:

- Прогнозування, електронний лист спам чи ні.
- Пухлина злоякісна чи доброякісна.
- Гриб отруйний чи їстівний.

При використанні логістичної регресії зазвичай задається поріг, який вказує, за якого значення приклад буде введено в один клас проти іншого класу. У завданні класифікації спаму може бути встановлено поріг 0,5, що призведе до того, що повідомлення електронної пошти з 50% або більшою ймовірністю буде класифікуваться як спам та будь-який електронний лист із ймовірністю менше 50%, класифікуватися як не спам. Хоча логістична регресія найкраще підходить для випадків двійкової класифікації, вона може бути застосована до задач класифікації класичних класів, завдань класифікації з трьома і більше класами, що досягається застосуванням стратегію один проти всіх. Нехай, є три різні класи, до яких можуть потрапляти екземпляри наборів даних, які можна розцінювати як три різні проблеми класифікації.

Наприклад, при тренуванні класифікатора лише на прикладах, що належать до класу A, порівняно з усіма прикладами, що належать до всіх інших класів. Потім необхідно зробити те саме, що і для класу B, і нарешті для класу C. Після того, як класифікатори навчилися відрізняти обраний ними клас від інших класів, запускаються три класифікатори на вхідних даних, і той клас класифікатора, який впевнений, що вибрав правильний клас для цього прикладу, саме такий клас приклажається до прикладу.

Логістична регресія – це потужний алгоритм машинного навчання, який використовує сигмоподібну функцію та найкраще працює над проблемами бінарної класифікації, хоча може бути використаний для задач класифікації класів за допомогою методу «один проти всіх». Логістична регресія (незважаючи на свою назву) не підходить для регресійних завдань. Для розуміння логістичної регресії необхідно застосувати її до інших

наборів даних, щоб побачити, у яких видах проблем класифікації вона добре впорається.

2.1.2. *K-найближчих сусідів*

KNN алгоритм – є однією з найбільш простих методик, що застосовуються в машинному навчанні. Це метод, якому віддають перевагу в галузі через його простоту використання та малий час розрахунку. KNN – модель, яка класифікує точки даних на основі точок, найбільш схожих на неї. Він використовує дані тесту, щоб скласти попередні знання про те, який некласифікований пункт слід класифікувати. Приклад на рис. 2.3.

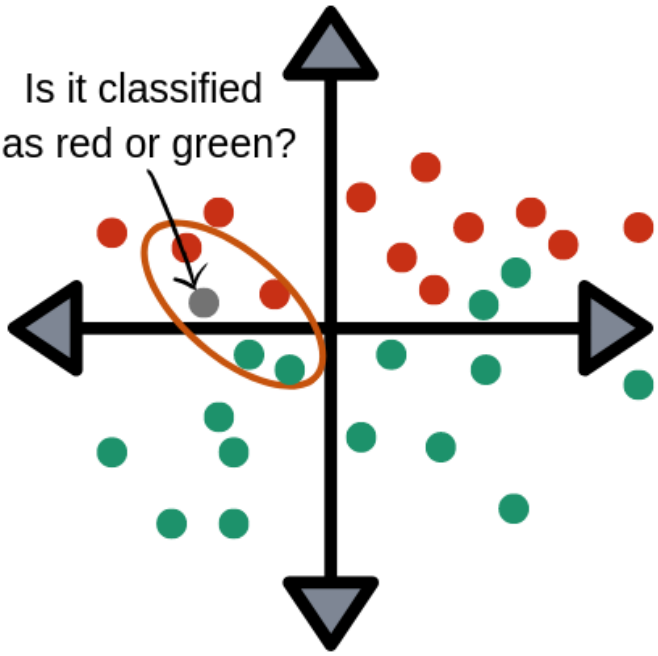


Рис. 2.3. Графічне зображення алгоритму KNN

Переваги:

- Простий у використанні.
- Швидке обчислення.
- Не робить припущень щодо даних.

Мінуси:

- Точність залежить від якості даних.
- Потрібно знайти оптимальне значення k (кількість найближчих сусідів).
- Бідні в класифікації точок дані на межі, де їх можна класифікувати так чи інакше.

KNN – алгоритм, який вважається як непараметричний, так і прикладом лінивого навчання. Непараметричне означає, що він не робить припущень. Модель повністю складається з наданих їй даних, а не припускаючи, що її структура є нормальною.

Ледаче навчання означає, що алгоритм не робить узагальнень. Це означає, що при використанні цього методу мало навчання. Через це всі дані тренінгу також використовуються при тестуванні під час використання KNN.

KNN часто використовується в простих системах рекомендацій, технологіях розпізнавання зображень та моделях прийняття рішень. Саме алгоритм компаній, таких як Netflix або Amazon, використовують KNN для того, щоб рекомендувати різні фільми для перегляду чи книги для придбання. Потім ці компанії введуть наявні ваші дані клієнтів і порівняють їх з іншими клієнтами, які переглянули подібні фільми або купили подібні книги. Потім ця точка даних буде класифікована як певний профіль на основі їх минулого за допомогою KNN. Рекомендовані фільми та книги залежатимуть від того, як алгоритм класифікує цю точку даних.

На рис. 2.4 зображено, як працює KNN, намагаючись класифікувати точку даних на основі заданого набору даних. Його порівнюють з найближчими його точками та класифікують, виходячи з того, які точки йому найближчі та найбільш схожі. Тут можна побачити, що точка X_j буде класифікована як W_1 (червона), або W_3 (зелена), залежно від її відстані від кожної групи точок.

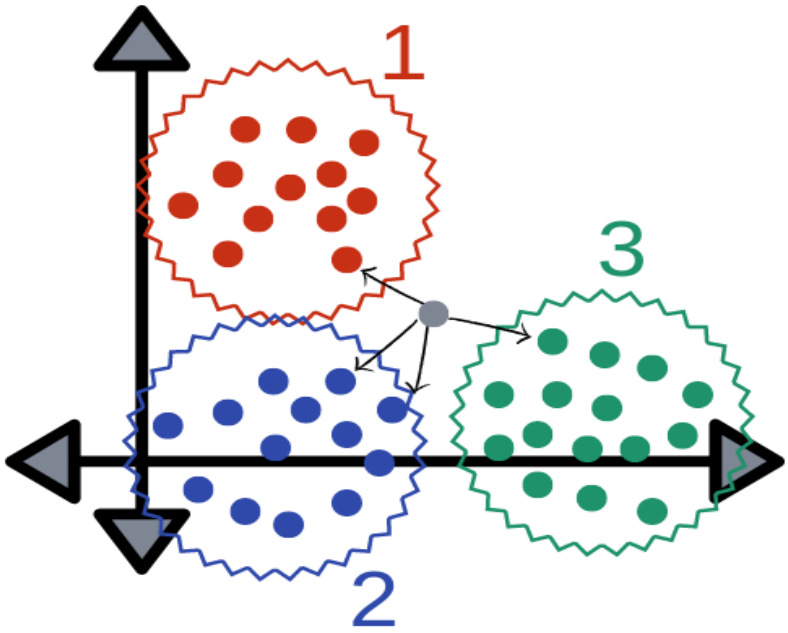


Рис. 2.4. Класифікація точки на зібранних даних алгоритму KNN

Як і майже все інше, KNN працює завдяки глибоко вкоріненим математичним теоріям, які він використовує. Реалізуючи KNN, перший крок – перетворення точок даних у вектори функцій або їх математичне значення. Потім алгоритм працює, знаходячи відстань між математичними значеннями цих точок. Найпоширеніший спосіб знайти цю відстань – евклідова відстань.

KNN виконує обчислення відстані між кожною точкою даних та тестовими даними. Потім він виявляє ймовірність того, що ці точки схожі з тестовими даними, і класифікує її, виходячи з того, які точки поділяють найбільші ймовірності. Візуалізація цього процесу продемонстрована на рис. 2.5.

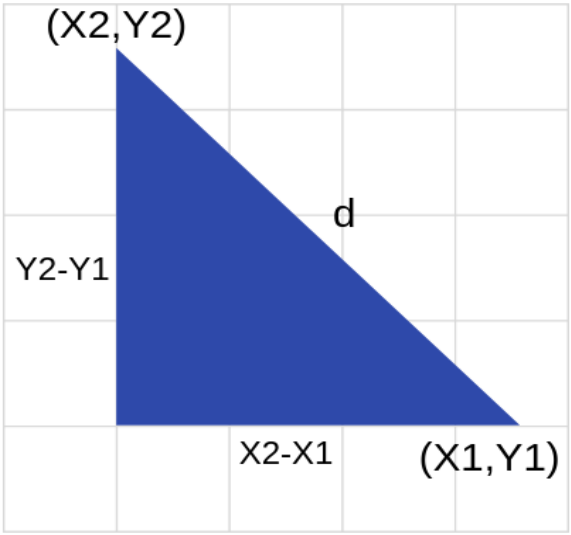


Рис. 2.5. Візуалізація роботи класифікатора

Якщо є набір даних з великою кількістю різних точок і точної інформації, це прекрасне місце для початку вивчення машинного навчання за допомогою KNN. Починаючи використовувати цей алгоритм, необхідно зауважити про:

- Спочатку необхідно знайти набір даних, з яким буде легко працювати, в ідеалі – з великою кількістю різних точок і розмічених даних.
- По-друге, необхідно з’ясувати, яка мова буде найпростішою для використання для вирішення проблеми.
- По-третє, необхідно провести дослідження.

Загалом KNN – це фундаментальний алгоритм машинного навчання, який є надійним з багатьох причин, таких як простота використання та швидкий час обчислення. Це хороший алгоритм, який можна використовувати, але він все ще має можливість вдосконалення та модифікації.

2.1.3. *Підтримка векторної машини*

Підтримка векторної машини (SVM) – це дискримінантний класифікатор, формально визначений роздільним гіперпланом. Іншими словами, з даними мічених навчальних даних (контрольоване навчання) алгоритм виводить оптимальний гіперплан, який класифікує нові приклади. У двомірному просторі цей гіперплан – це лінія, що розділяє площину на дві частини, де в кожному класі лежать обидві сторони.

Припустимо, присвоєно графік двох класів міток на графіку, як показано на рис. 2.6.

Рис. 2.6. Графік міток класів

SVM справедливо розділяє два класи (рис. 2.7). Будь-яка точка, яка ліворуч від лінії, належить до класу чорних кругів, а справа належить до класу синього квадрата. Поділ занять. Це те, що робить SVM. Він виявляє лінію або гіперплощину (у багатовимірному просторі, що розділяє класи).

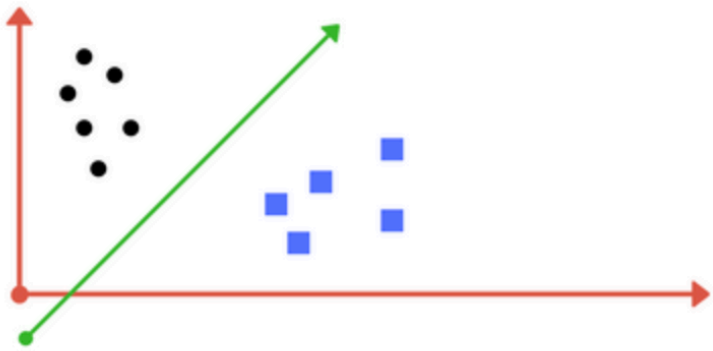


Рис. 2.7. Поділ на класи

Зрозуміло, що немає лінії, яка могла б розділити два класи в площині $x - y$. Необхідно застосувати перетворення і додаємо ще один вимір, вісь z. Допустимо значення точок на площині z, $w = x^2 + y^2$. У цьому випадку можна маніпулювати нею як відстань точки від z-початку. Тепер, якщо побудувати графік по осі z, видно чіткий поділ і можна провести лінію (рис. 2.8).

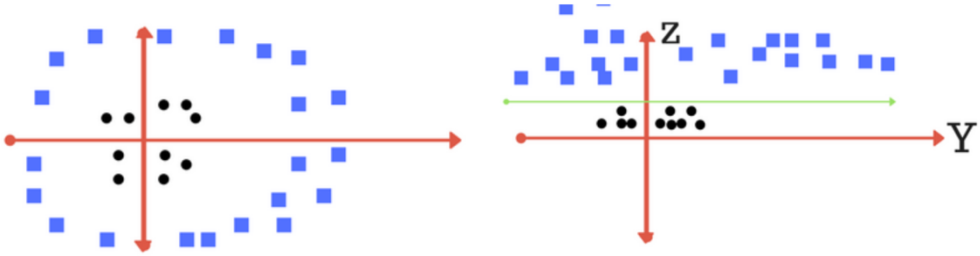


Рис. 2.8. Введення площини Z

Коли перетворимо цю лінію в початкову площину, вона відображається до кругової межі. Ці перетворення називаються ядрами (зображено на рис. 2.9).

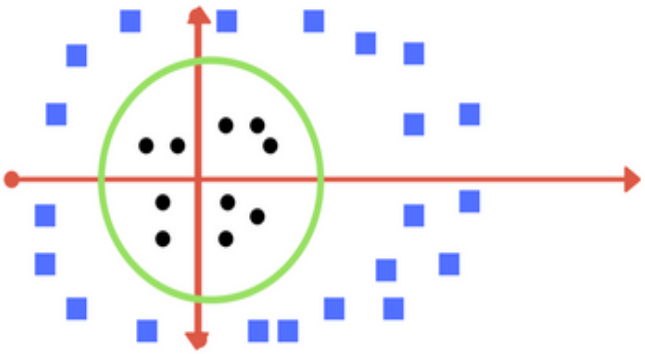


Рис. 2.9. Поділ на ядра

У реальному застосуванні пошук ідеального класу для мільйонів наборів даних при навчанні вимагає багато часу. Це називається параметром регуляризації. Далі визначимо два терміни, параметр регуляризації та гамма. Це налаштування параметрів у класифікаторі SVM. Зважаючи на те,

що можна досягти значної нелінійної класифікаційної лінії з більшою точністю за розумну кількість часу, можна підвищити точність SVM шляхом налаштування цих параметрів. Ще один параметр – ядро. Він визначає, чи необхідно використати лінійний поділ.

Вивчення гіперплану в лінійному SVM здійснюється шляхом перетворення задачі за допомогою лінійної алгебри. Для лінійного ядра рівняння для прогнозування нового введення з використанням крапкового добутку між входом *x* та кожним вектором підтримки.

Параметр регуляризації (часто його називають параметром C у бібліотеці sklearn python) повідомляє оптимізацію SVM, наскільки необхідно уникнути неправильної класифікації кожного прикладу навчання. Для великих значень C оптимізація обирає гіперплан з меншою границею, якщо цей гіперплан виконає кращу роботу щодо правильного класифікації всіх навчальних балів. І навпаки, дуже невелике значення C призведе до того, що оптимізатор буде шукати гіперплан, що розділяє великі запаси, навіть якщо гіперплан неправильно класифікує більше балів.

Параметр гамма визначає, наскільки далеко досягає вплив одного прикладу тренувань, з низькими значеннями, що означають далеко, а високі значення означають близько. Іншими словами, при низькій гаммі точки, віддалені від правдоподібної лінії відокремлення, враховуються при обчисленні лінії відокремлення. Там, де висока гамма означає, точки, близькі до правдоподібної лінії, враховуються при обчисленні (рис. 2.10).

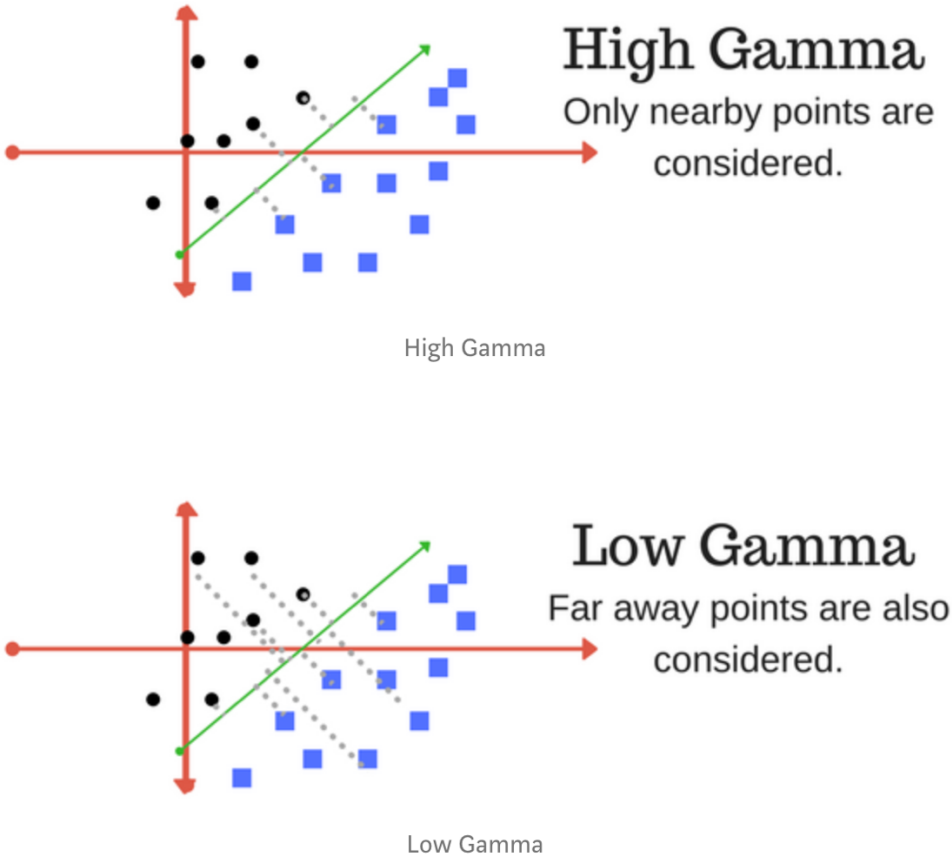


Рис. 2.10. Вплив параметру гамма

І, нарешті, остання, але дуже важлива характеристика класифікатора SVM. SVM до ядра намагається досягти хорошого запасу в полі. Поле – це розділення лінії до найближчих точок класу. Хороший запас – той, де цей поділ більший для обох класів. Зображення, наведені на рис. 2.11, наводять наочний приклад хорошого та поганого запасу. Хороший запас дозволяє балам бути у відповідних класах без переходу на інший клас.

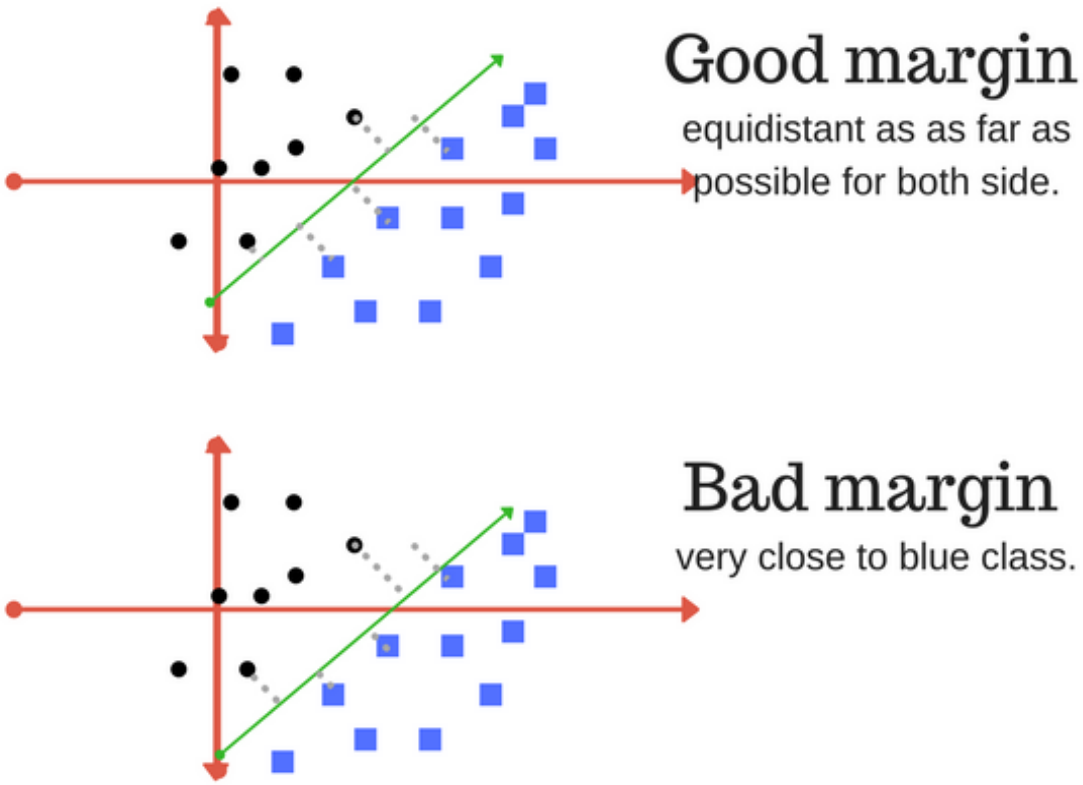


Рис. 2.11. Вплив параметру margin

2.2. Узагальнений огляд етапів методу

Запропонований метод складається з таких етапів:

- Пошук і використання даних – в одній з двох форм: структуровані або неструктуровані. Усередині кожної з них є статичні або потокові дані.
    - Структуровані дані – уявіть собі таблицю рядків і стовпців, електронну таблицю транзакцій клієнтів, базу даних записів пацієнтів. Стовпці можуть бути числовими начебто середньої частоти серцевих скорочень, категоріальним начебто статі людини або порядковими начебто інтенсивності болю в грудях.
    - Неструктуровані дані – все, що не може бути відразу вміщено в формат рядка і стовпця: зображення, аудіофайли, текст на

природній мові. Статичні дані – існуючі історичні дані, які навряд чи зміняться. Хороший приклад - історія покупок.

- Потокові дані – дані, які постійно оновлюються, старі записи можуть бути змінені, нові записи постійно додаються. Варто відзначити, що тут можуть бути перехресні типи даних. Статична структурована інформаційна таблиця може мати стовпці, які містять текст на природній мові і фотографії, і при цьому постійно оновлюватися. Для прогнозування серцевих захворювань один стовпець може містити підлогу, інший – середній серцевий ритм, третій – середній артеріальний тиск, четвертий – інтенсивність болю в грудях. Для прикладу зі страховим відшкодуванням один стовпець в таблиці може бути з текстом, відправленим клієнтом в заяві, інший стовпець може бути з зображенням, яке він відправив разом з текстом, і в останньому стовпчику – результат позову. Ця таблиця щодня оновлюється новими заявками або зміненими результатами старих заявок.

- Оцінка результатів – існують різні метрики оцінки для задач класифікації, регресії і рекомендацій. Модель з точністю 95% може здатися досить хорошою для передбачення винного в страховому позові. Але для прогнозування серцево-судинних захворювань, ймовірно, необхідні більш точні результати. Є й інші речі, які потрібно взяти до уваги при класифікації завдань.
    - Хибне негативне спрацьовування – модель прогнозує негативний варіант, а насправді він позитивний. У деяких випадках, таких як прогнозування спаму в електронній пошті, помилкові спрацьовування не так вже й страшні. Але буде набагато гірше, якщо система комп'ютерного зору для автомобілів з автопілотом не розпізнає пішохода, коли насправді він є.

- Хибне позитивне спрацьовування – модель передбачає позитивний варіант, а насправді він негативний. Якщо людині передбачать хвороба серця, від якої вона насправді не страждає, може здатися не таким вже страшним, але це негативно впливає на спосіб життя людини або встановлює для нього план лікування, в якому вона не потребує.
- Істинне негативне спрацьовування – модель прогнозує негативний варіант, який насправді таким і є.
- Істинне позитивне спрацьовування – модель передбачає позитивний варіант, який насправді таким і є. Це також добре.
- Точність – визначається, яка частка позитивних прогнозів була правильною Модель, яка не дає помилкових спрацьовувань, має точність 1.0.
- Повнота – частка фактичних позитивних варіантів, яка була передбачена правильно. Модель, яка не дає помилкових негативних варіантів, має відгук 1.0.
- Оцінка F1 – поєднання точності і повноти. Чим ближче до 1.0, тим краще.
- Крива робочих характеристик приймача (ROC) і площа під цією кривою (AUC) – крива ROC є графік, що порівнює співвідношення справжніх позитивних і хибних позитивних варіантів. Метрика AUC – це площа під кривою ROC. Модель, чиї прогнози на 100% невірні, має AUC 0.0, а модель, чиї прогнози є 100% правильними, має AUC 1.0.

Проблеми з документами чи описами складніше перевірити експериментально. Один із способів зробити це – взяти частину даних і заховати їх. Коли модель побудована, використовуйте її, щоб передбачити рекомендації для прихованих даних і подивитися, як вони шикуються. Припустимо, ви намагаєтеся рекомендувати покупцям продукти в своєму

інтернет-магазині. У вас є архівні дані про покупки за 2010-2019 рр. Ви можете побудувати модель на основі даних за 2010-2018 рр., А потім використовувати її для прогнозування покупок в 2019 р Тоді це стає проблемою класифікації, тому що ви намагаєтеся визначити, чи може хтось купити що-небудь. Однак традиційні метрики класифікації не кращий варіант для задач з рекомендаціями. Точність і повнота не мають поняття порядку. Якщо ваша модель машинного навчання повернула список з 10 рекомендацій, які будуть показані клієнту на вашому веб-сайті, ви б хотіли, щоб кращі з них відображалися першими, вірно? Точність *k* – те саме, що і звичайна точність, проте ви вибираєте відсікання *k* варіантів. Наприклад, точність 5 означає, що вам важливі тільки 5 кращих рекомендацій. У вас може бути 10 000 продуктів, але ви не можете рекомендувати їх всім своїм клієнтам. Для початку у вас може не бути точної цифри для кожного з них. Але знаючи, на які метрики ви повинні звертати увагу, ви отримаєте уявлення про те, як оцінити ваш проект машинного навчання.

- Особливості – не всі дані однакові. І коли хтось посилається на особливості, вони в свою чергу посилаються на різні види даних в даних. Три основних типи особливостей – категоріальні, безперервні (або чисельні) і похідні:
    - Категоріальні особливості – наприклад, у проблеми з серцем – це стать пацієнта. Або для інтернет-магазину – зробив хтось покупку чи ні.
    - Безперервні (або чисельні) особливості – числові значення на зразок середньої частоти серцевих скорочень або число входів в систему.
    - Похідні особливості – особливості, які створюються з даних. Часто згадуються як конструювання ознак. Конструювання ознак (особливостей) – це коли експерт предметної області бере свої знання і кодує їх в дані. Можна об'єднати кількість входів в систему з тимчасовими мітками, щоб створити

особливість і дати їй назву «час з моменту останнього входу». Варто відзначити також деякі важливі аспекти, які потрібно пам'ятати, коли мова йде про особливості.

- Зберігати їх необхідно однаковими під час експериментів (навчання) і тестування – модель машинного навчання повинна бути навчена на особливостях, максимально наближених до того, для чого вона буде використовуватися в реальній системі.
- Робота з експертами певної предметної області – краще вибрати особливості з найбільшим обхватом, ті, де безліч зразків мають дані.
- Ідеально значить зламано – якщо у вашій моделі ідеальна продуктивність, то, ймовірно, допущено витік особливостей. Це означає, що дані, на яких навчалася модель, використовуються для їх перевірки. Жодна модель не ідеальна. Ці особливості спрощені і не повинні бути точними. Але саме це буде використовуватись, щоб побачити, чи може машинне навчання покращитися чи ні.

- Експерименти – цей крок включає в себе всі інші кроки. Оскільки машинне навчання – це ітеративний процес, потрібно переконатися, що експерименти ефективні. Головна мета – звести до мінімуму час між автономними експериментами і онлайн-експериментами. Автономні експерименти – це кроки, коли проект ще не орієнтований на клієнта. Онлайн-експерименти проводять, коли модель машинного навчання знаходиться в продакшині. Всі експерименти повинні проводитися на різних ділянках даних.
  - Набір даних для навчання – використовується цей набір для навчання моделі, рекомендований об'єм 70 чи 80% даних.

- Набір даних для перевірки та розробки – використовується цей набір для налантування моделі, рекомендований об'єм – 10 чи 15% даних.
- Набір тестових даних – використовується цей набір для тестування і порівняння моделей, рекомендований обсяг – 10 чи 15% даних. Ці співвідношення можуть незначно коливатиметься в залежності від завдання і наявних даних.

Низька продуктивність за даними навчання означає, що модель не вивчена належним чином. Низька продуктивність на тестових даних означає, що ваша модель погано узагальнюється. Модель перенавчитися на даних для тренування. Низька продуктивність після розгортання (в реальному світі) означає, що існує різниця між тим, на чому навчалась і тестувалась модель, і тим, що відбувається на практиці.

2.3. Висновки до розділу 2

У даному розділі розглянуто опис алгоритмів машинного навчання, обраних для реалізації автоматизованого тегування текстових документів, розглянуто можливості даних моделей, щоб реалізувати ансамбль моделей. Проведено огляд алгоритмів, які дозволяють пришвидшити пошук тегів у тестових документах. Запропоновано спосіб пошуку тегів у текстових даних, завдяки ансамблю моделей машинного навчання для роботи з текстом. Наведено вхідні дані та результат роботи алгоритмів. Зазначено перелік та детальний опис кроків побудови алгоритму, що дозволяють досягти необхідного результату. Можливі реалізації даних етапів запропоновано в наступному розділі.

3. ОСОБЛИВОСТІ РЕАЛІЗАЦІЇ СПОСОБУ АВТОМАТИЗОВАНОГО ТЕГУВАННЯ ТЕКСТОВИХ ДОКУМЕНТІВ

3.1. Обґрунтування вибору засобів для реалізації способу

В попередньому розділі наведено перелік етапів, з яких складається спосіб. До цих етапів належать:

- Пошук і використання даних.
- Оцінка результатів.
- Визначення особливостей.
- Експерименти.

Для реалізації наведеного вище методу обрано мову програмування Python та бібліотеку Scikit learn.

Python – є мовою загального призначення високого рівня, яка є орієнтованою на підвищення ефективності у розробці. У стандартному наборі бібліотек перелічений високих обсяг корисних можливостей.

Python підтримує декілька парадигм програмування:

- Структурне.
- Об'єктно-орієнтоване.
- Функціональне.
- Імперативне.
- Аспектно-орієнтоване.

Основні архітектурні риси – динамічна типізація, автоматичне керування пам'яттю, механізм обробки виключень, підтримка багатопоточних обчислень і зручні високорівневі структури даних. Код у Python організовується у функції та класи, які можуть об'єднуватися в модулі (що в свою чергу можуть бути об'єднані в пакети).

Python використовує динамічну типізацію. Всі значення є об'єктами, в тому числі функції, методи, модулі, класи.

Мова володіє чітким і послідовним синтаксисом, перевагами є модульність та масштабованість, завдяки чому вихідний код програм легко зрозуміти.

Дизайн мови Python побудований навколо об'єктно-орієнтованої моделі програмування. Реалізація ООП в Python є елегантною, потужною і добре продуманої, але разом з тим досить специфічною в порівнянні з іншими об'єктно-орієнтованими мовами.

- Класи є одночасно об'єктами з усіма нижче наведеними можливостями.
- Спадкування, в тому числі множинне.
- Поліморфізм (всі функції віртуальні).
- Інкапсуляція (два рівні – загальнодоступні та приховані методи і поля). Особливість - приховані члени доступні для використання і позначені як приховані лише особливими іменами.
- Спеціальні методи, що керують життєвим циклом об'єкта: конструктори, деструктори, розподільники пам'яті.
- Перевантаження операторів (всіх, окрім is, '.', '=' I символьних логічних).
- Властивості (імітація поля за допомогою функцій).
- Управління доступом до полів (емуляція полів і методів, частковий доступ, і т. п.).
- Методи для управління найбільш поширеними операціями (истинностное значення, len (), глибоке копіювання, сериализация, ітерація по об'єкту)
- Метапрограмування (управління створенням класів, тригери на створення класів, та ін.)
- Повна інтроспекція.
- Класові і статичні методи, класові поля.
- Класи, вкладені в функції і класи.

Дизайн мови Python побудований навколо об'єктно-орієнтованої моделі програмування. Реалізація ООП у Python є елегантною, потужною і добре продуманої, але разом з тим досить специфічної порівняно з іншими об'єктно-орієнтованими мовами.

Багата стандартна бібліотека є однією з привабливих сторін Python. Присутні засоби для роботи з багатьма мережевими протоколами і форматами Інтернету, наприклад, модулі для написання HTTP-серверів і клієнтів, для розбору і створення поштових повідомлень, для роботи з XML і т.п. Набір модулів для роботи з операційною системою дозволяє писати кросплатформні додатки. Існують модулі для роботи з регулярними виразами, текстовим кодуванням, мультимедійними форматами, криптографічними протоколами, архівами, серіалізації даних, підтримка юніт-тестування.

Бібліотека Scikit learn – найкращий вибір з бібілотек, які використовуються для завдань, з використанням машинного навчання. Вона надає широкий вибір алгоритмів навчання з учителем і без вчителя. Навчання з учителем передбачає наявність розміченого набору даних, в якому передбачене значення показників. У той час як навчання без вчителі не передбачає наявності розмітки в датасета – потрібно навчитися отримувати корисну інформацію з довільних даних. Одне з основних переваг бібліотеки полягає в тому, що вона працює на основі декількох поширених математичних бібліотек, і легко інтегрує їх один з одним. Ще однією перевагою є широка спільнота і докладна документація. Scikit learn широко використовується для промислових систем, в яких застосовуються алгоритми класичного машинного навчання, для досліджень, а так само для новачків, які тільки робить перші кроки в області машинного навчання.

Для своєї роботи, scikit learn використовує такі популярні бібліотеки:

- NumPy – математичні операції і операції над тензорами
- SciPy – науково-технічні обчислення
- Matplotlib – візуалізація даних

- IPython – інтерактивна консоль для Python
- SymPy – символьна математика
- Pandas – обробка, маніпуляції і аналіз даних

До завдань бібліотеки не входить завантаження, обробка, маніпуляція даними і їх візуалізація. З цими завданнями відмінно справляються бібліотеки Pandas і NumPy. Scikit learn спеціалізується на алгоритмах машинного навчання для вирішення завдань навчання з учителем: класифікації (прогноз ознаки, безліч допустимих значень якого обмежена) і регресії (прогноз ознаки з речовими значеннями), а також для задач навчання без учителя: кластеризації (розбиття даних по класах , які модель визначить сама), зниження розмірності (подання даних в просторі меншої розмірності з мінімальними втратами корисної інформації) і детектування аномалій.

Бібліотека реалізує наступні основні методи:

- Лінійні: моделі, завдання яких побудувати розподіл (для класифікації) або апроксимацію (для регресії) гіпер площину.
- Метричні: моделі, які обчислюють відстань по одній з метрик між об'єктами вибірки, і приймають рішення в залежності від відстані (*k*-найближчих сусідів).
- Дерева рішень: навчання моделей, що базуються на безлічі умов, оптимально обраних для вирішення завдання.
- Ансамблеві методи: методи, засновані на деревах рішень, які комбінують міць безлічі дерев, і таким чином підвищують їх якість роботи, а також дозволяють проводити відбір ознак (бустінг, беггінг, випадковий ліс, голосування).
- Нейронні мережі: комплексний нелінійний метод для задач регресії і класифікації.
- SVM: нелінійний метод, який навчається визначати межі прийняття рішень.

- Наївний Байєс: прямий розподіл моделювання для задач класифікації.
- PCA: лінійний метод зниження розмірності і відбору ознак
- *t*-SNE: нелінійний метод зниження розмірності.
- *k*-середніх: найпоширеніший метод для кластеризації, требущіх на вхід число кластерів, за якими повинні бути розподілені дані.
- Крос-валідація: метод, при якому для навчання використовується весь датасета (на відміну від розбиття на вибірки train та test), проте навчання відбувається багаторазово, і в якості валідаційної вибірки на кожному кроці виступають різні частини датасета. Підсумковий результат уявляють собою усереднення отриманих результатів.
- Grid Search: метод для знаходження оптимальних гіперпараметрів моделі шляхом побудови сітки із значень гіперпараметрів і послідовного навчання моделей з усіма можливими комбінаціями гіперпараметрів для сітки.

Для побудови ансамблю моделей машинного навчання для автоматизованого тегування текстових документів, необхідно використати готовий набір даних для навчання. Як основу для аналізу обрано тексти описів з сайтів вакансій для IT-розробників. Ці тексти характеризуються наявністю великого набору прихованих категорій, які залежать від локації та часу публікації оголошення про роботу. Через це класичні моделі з оброблення текстових даних матимуть значні похибки у результатах класифікації, адже вони не враховують усіх особливостей опису вакансій.

Як зазначено вище, існуючі на сьогоднішній день способи автоматизованого тегування текстових документів не є достатньо ефективними і характеризуються низькою точністю класифікації оброблюваних текстів.

В якості критерія ефективності в даній роботі використовуватимемо міру F1, яка у статистичному аналізі двійкової класифікації є мірою точності визначення категорій, які присутні в тексті .

F-оцінка враховує як точність *p*, так і похибку *r* визначення категорій для обчислення показника відхилення від правильного результату: *p* – кількість правильних позитивних результатів, поділена на кількість усіх позитивних результатів, повернутих класифікатором, *r* – кількість правильних позитивних результатів, поділена на кількість усіх відповідних зразків (усі зразки, які повинні були бути визначені як позитивні). Оцінка F1 – це середнє значення похибки та точності, де оцінка F1 досягає свого найкращого значення в 1 (ідеальна точність та відкликання) і найгіршого при 0.

Для кожного рішення задачі з використанням машинного навчання спочатку необхідно провести роботу з очищення вхідних даних, щоб отримати повний набір даних для тренування моделі. Для виконання завдань даного дослідження в якості джерела вхідних даних обрано сайти з публікаціями IT-вакансій, адже ті містять фіксований набір категорій.

Оскільки не у кожній публікації на сайтах для пошуку роботи є інформація про зарплату, потрібно було зібрати дані з тисяч сторінок, щоб після пошуку вакансій була принаймні 1000 публікацій з даними про фіксовані зарплати.

Для аналізу обрано лише ці дванадцять тегів серед текстових даних, які використовуються у класифікації:

- “part-time-job”;
- “full-time-job”;
- “hourly-wage”;
- “salary”;
- “associate-needed”;
- “bs-degree-needed”;
- “ms-or-phd-needed”;
- “licence-needed”;
- “1-year-experience-needed”;
- “2-4-years-experience-needed”;

- “5-plus-years-experience-needed”;
- “supervising-job”.

Після того, як етап парсингу даних було виконано, з’явилась необхідність врахувати, що параметр “City” посилається на список усіх міст, які вказано на сайтах з публікаціями вакансій. Тому запропоновано обрати 1000 сторінок з оголошеннями про вакансії для найбільших міст і лише 50 – для найменших, оскільки у них є дуже мало публікацій про робочі місця, а тому й виникає більше проблем з класифікацією цих даних.

Також при аналізі вхідних даних враховано дати публікацій вакансій, адже вони мають вагомий вплив на результати визначення категорії заробітної плати не вказаної в описі вакансії, а отже і на актуальність вакансії в цілому для конкретного робітника.

Після того, як всі дані з сайтів було зібрано, з них було створено два різних набори даних:

- базовий набір – містить основну інформацію, очищену від усіх назв посад та деяких інших полів, які знаходяться в публікації;
- векторизований набір – містить кожне слово, яке з’явилося в будь-якій публікації, перетворене на стовпець в таблиці даних, де зберігається значення кількості разів, скільки слово з’явилося загалом в описі вакансії (в основному використовується як резервне копіювання вхідного тексту).

Для того, щоб зібрати базовий набір даних, запропоновано створити у моделі кілька нових параметрів для визначення категорій, які характеризують цікаві особливості назви вакансій чи резюме:

- Job_Kind: є визначеним параметром за замовчуванням, який використовується для оброблення веб-сторінок сайтів вакансій, а також для зберігання даних як додаткових властивостей (або важливих додаткових категорій) для створеного набору даних. Job_Kind вказує на тип договору, який може бути («Повний

робочий день», «Постійний», «Контракт», «Тимчасовий», «Стажування», «За сумісництвом» або «Учень»).

- Стажування: параметр, створений з назви та опису роботи, в яких вказано про підлеглого та наставника чи відсутність необхідності деяких вказаних очікувань від робітника.

На основі використання базового і векторизованого наборів даних надалі проведено аналіз для автоматизованого тегування описів вакансій.

3.2. Огляд програмної реалізації розробленого методу

З описаного в попередніх розділах очевидно, що реалізація запропонованого вище способу вимагає зваженого проектування архітектури системи. Розглянемо сучасні архітектурні підходи, які можуть бути використані для реалізації відповідного програмного забезпечення.

На початковому етапі аналізу даних використовуємо всі моделі зі списку, визначеного вище, але одну з них обрано як базову - логістичну регресію з перехресною валідацією, щоб знайти початкове значення похибки на тестових даних і вирішити, з якими моделями надалі необхідно продовжувати дослідження. Є деякі чіткі властивості текстів опису вакансій, які визначають, чи можна віднести посаду до тієї чи іншої категорії, яку користувач буде вводити при пошуку певного типу робот и зображено на рис. 3.1. Для роботи алгоритмів машинного навчання потрібно перетворити текстові файли в числові вектори функцій. Для нашого прикладу будемо використовувати модель пакету слів. Коротко ми сегментуємо кожен текстовий файл на слова (для розбиття англійською мовою на пробіл) і підраховуємо кількість разів, коли кожне слово трапляється в кожному документі, і, нарешті, присвоюємо кожному слову цілий ідентифікатор. Кожне унікальне слово в нашому словнику буде відповідати функції (описової функції). Scikit learn має компонент високого рівня, який створить функціональні вектори для нас "CountVectorizer".

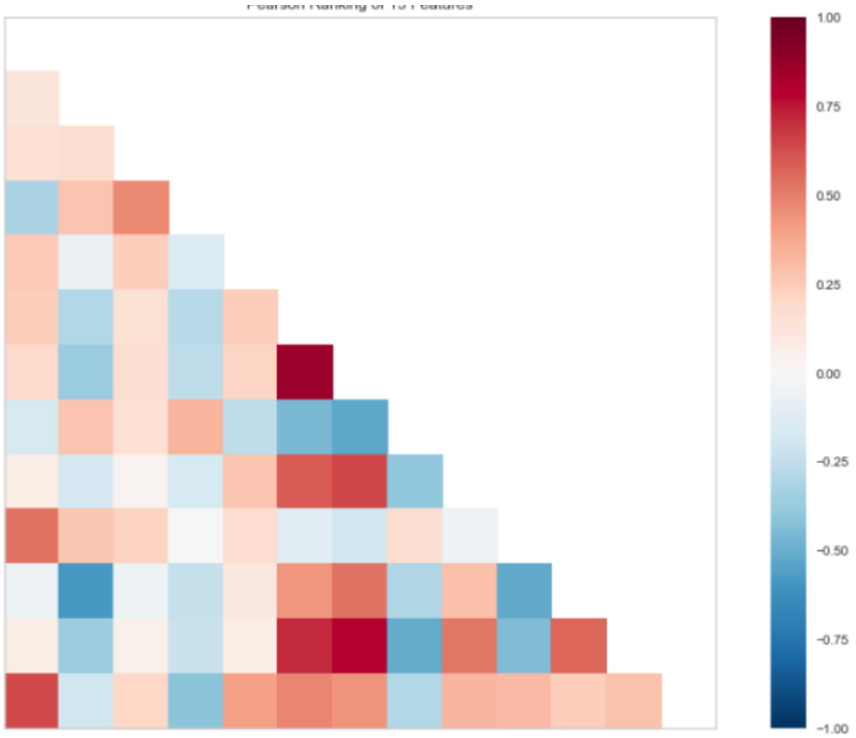


Рис. 3.1. Діаграма визначення важливості міток серед векторизованого набору даних з бібліотеки Scikit learn

Текстові файли – це фактично ряд слів (упорядкований). Для роботи алгоритмів машинного навчання потрібно перетворити текстові файли в числові вектори функцій. Для нашого прикладу будемо використовувати модель пакету слів. Коротко ми сегментуємо кожен текстовий файл на слова (для розбиття англійською мовою на пробіл) і підраховуємо кількість разів, коли кожне слово трапляється в кожному документі, і, нарешті, присвоюємо кожному слову цілий ідентифікатор. Кожне унікальне слово в нашому словнику буде відповідати функції (описової функції). Scikit learn має компонент високого рівня, який створить функціональні вектори.

Виконуючи метод "count_vect.fit_transform (dvadeset_train.data)", вивчається словник, і повертається матриця Document-Term. [n_samples, n_features]:

- TF – просто підрахунок кількості слів у кожному документі має 1 випуск: він дасть більше зважування довшим документам, ніж коротшим. Щоб цього уникнути, у кожному документі

використано частоту (TF – термінова частота), тобто кількість входжень слова поділена на загальну кількість слів.

- TF-IDF – можна навіть зменшити вагу більш поширених слів, таких як (the, is, тощо), які зустрічаються у всіх документах. Це називається TF-IDF, тобто термін – відношення числа входжень деякого слова до загальної кількості слів документа.

Як показує практика, спочатку краще поглянути на самі дані, а тільки потім провести їх очищення. Чистий датасета дозволить моделі вивчити значимі ознаки і не перенавчитися на нерелевантні шумі. Далі необхідно виконати такі операції при очищенні даних:

Видалити всі нерелевантні символи (наприклад, будь-які символи, що не належать до цифро-буквеним).

- Токенізіровать текст, розділивши його на індивідуальні слова.
- Видалити нерелевантні слова.
- Перевести всі символи в нижній регістр для того, щоб слова «привіт», «Привіт» і «ПРИВІТ» вважалися одним і тим же словом.
- Розглянути можливість суміщення слів, написаних з помилками, або мають альтернативне написання (наприклад, «круто» / «круть» / «крууууто»)
- Розглянути можливість проведення лематизації, відомості різних форм одного слова до словникової форми.

Після того, як будуть виконані ці кроки і необхідно виконати перевірку на додаткові помилки, після чого можна починати використовувати чисті, помічені дані для навчання моделей.

У словнику з описів вакансій міститься близько 20000 слів. Це означає, що кожне речення буде відображено вектором довжиною 20000. Цей вектор буде містити переважно нулі, оскільки кожне речення містить лише малу підмножину з нашого словника. Для того, щоб з'ясувати, захоплюють наші векторні уявлення (embeddings), релевантну нашого завдання інформацію), варто спробувати візуалізувати їх і подивитися,

наскільки добре розділені ці класи. Оскільки словники зазвичай є дуже великими і візуалізація даних на 20000 вимірювань неможлива, підходи на кшталт методу головних компонент (PCA) допомагають спроектувати дані на два виміри на рис. 3.2.

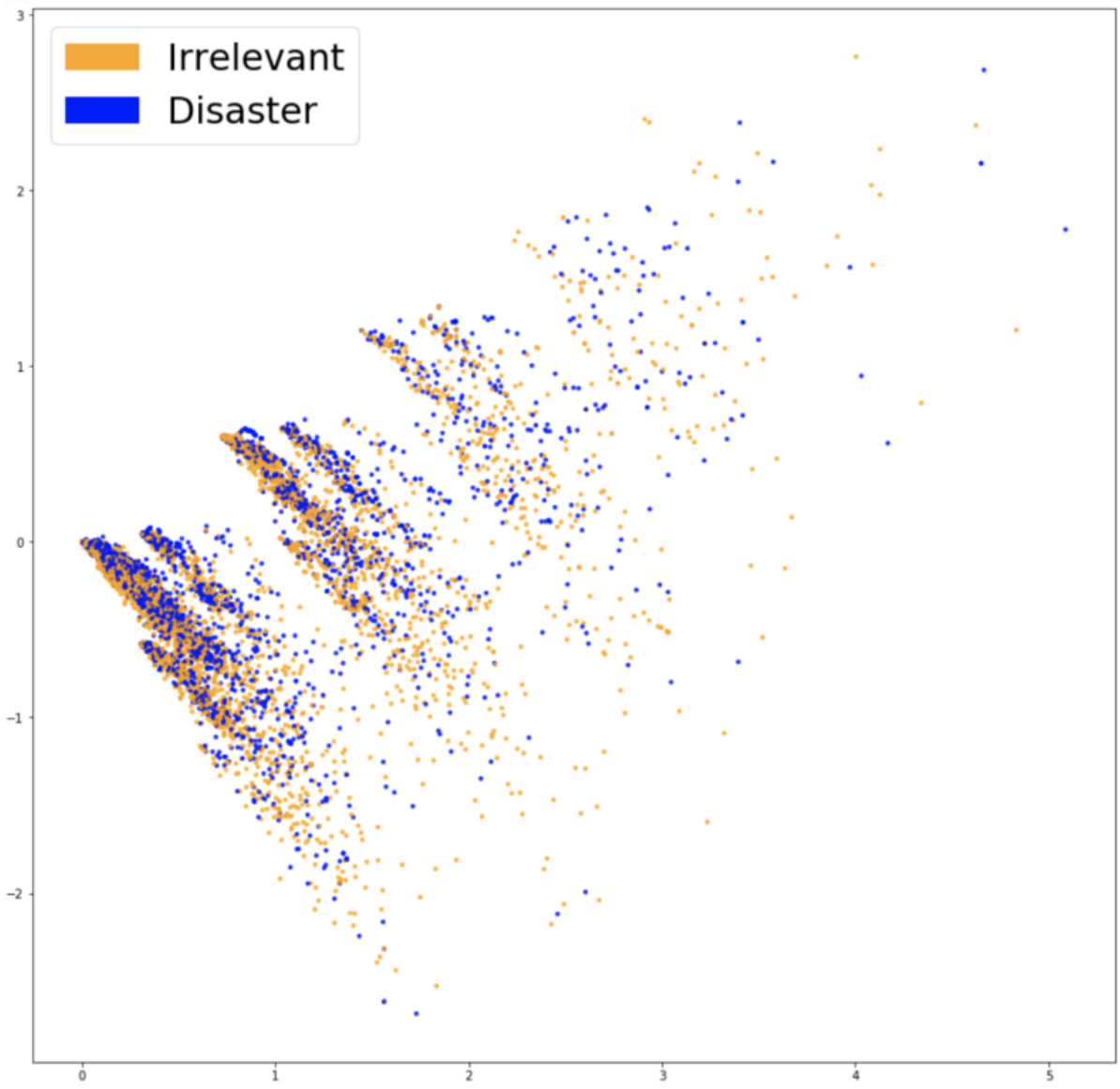


Рис. 3.2. Візуалізація векторних уявлень для «мішка слів»

Щоб допомогти моделі сфокусуватися на значущих словах, використано скоринг TF-IDF (Term Frequency, Inverse Document Frequency) поверх нашої моделі «мішка слів». TF-IDF зважує на підставі того, наскільки вони рідкісні в нашому датасета, знижуючи в пріоритеті слова, які

зустрічаються дуже часто і просто додають шум. Нижче наводиться проекція методу головних компонент, що дозволяє оцінити нове уявлення на рис. 3.3.

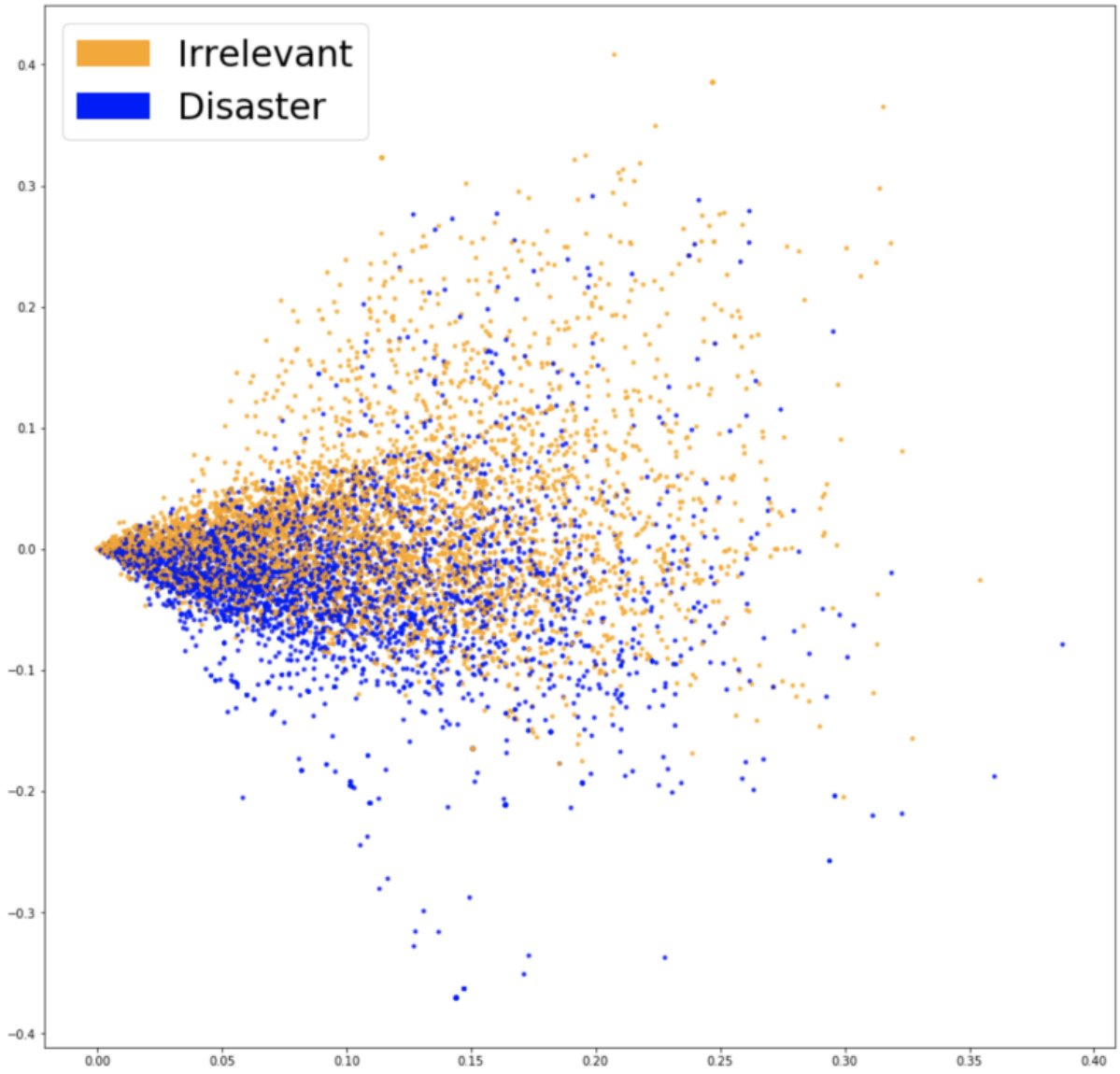


Рис. 3.3. Візуалізація векторного уявлення із застосуванням TF-IDF

Спостерігається більш чіткий поділ між двома кольорами. Це свідчить про те, що класифікатору має стати простіше розділити обидві групи.. Навчивши іншу логістичну регресію на наших нових векторних уявленнях, ми отримаємо точність в 76,2%. Дуже незначне поліпшення. Якщо отриманий результат по цій частині став краще, і не даємо моделі «шахраювати», то можна вважати цей підхід удосконаленням.

Швидким способом отримати вкладення пропозицій для класифікатора буде усереднення оцінок Word 2 Vec для всіх слів в пропозиції. Це все той же підхід, що й з «мішком слів» раніше, але на цей раз втрачаємо тільки синтаксис нашої пропозиції, зберігаючи при цьому семантичну (смислове) інформацію.

Векторні подання пропозицій в Word2Vec. Візуалізація наших нових векторних уявлень після використання перерахованих технік на рис. 3.4.

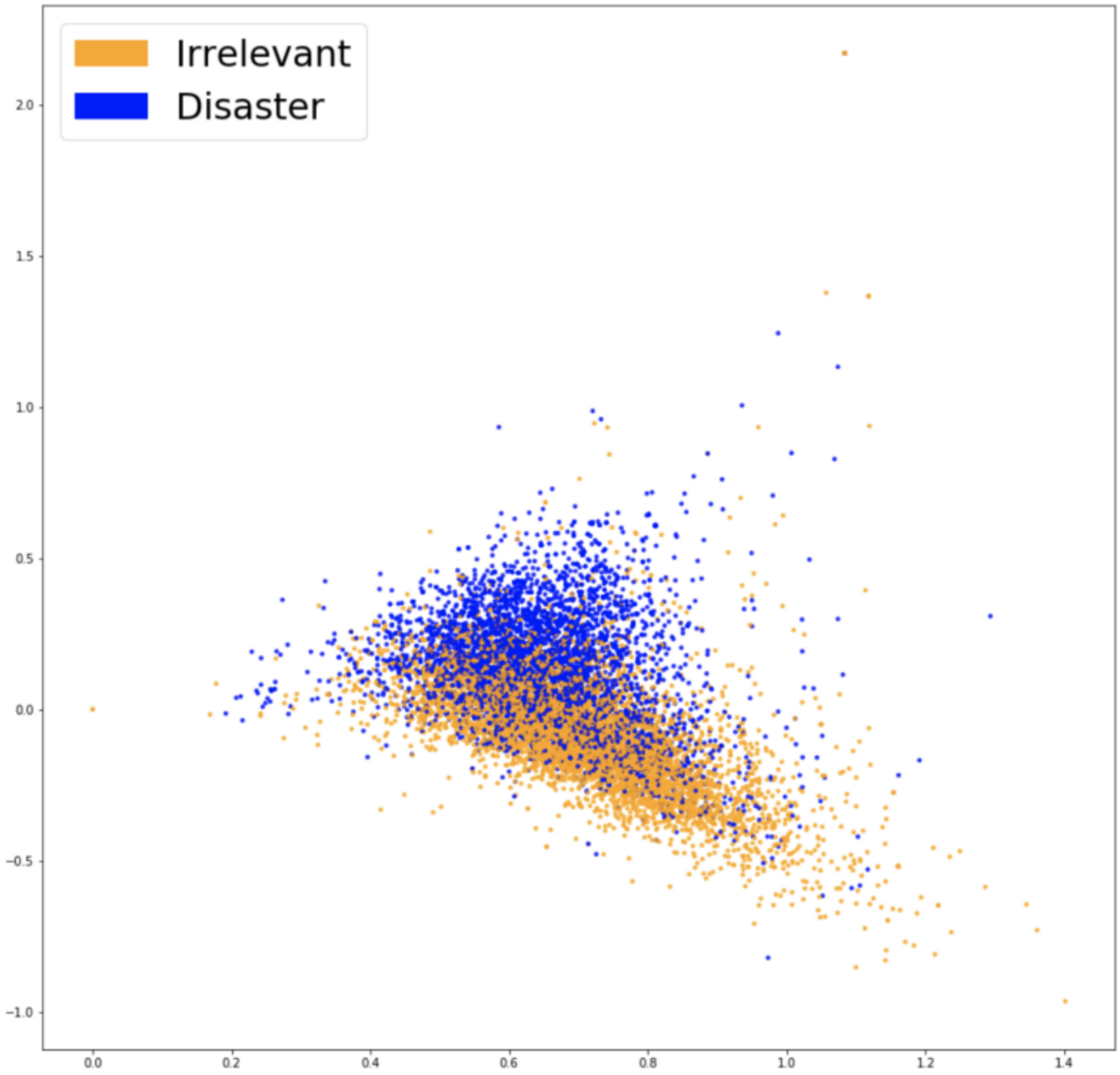


Рис. 3.4. Візуалізація векторних уявлень Word2Vec

Тепер дві групи кольорів виглядають розділеними ще сильніше, і це повинно допомогти класифікатором знайти відмінність між двома класами.

Після навчання тієї ж моделі в третій раз (логістична регресія), отримуємо точність в 77,7% і це наш найкращий результат на даний момент.

В бібліотеці Scikit-learn є модуль VotingClassifier, який дозволяє використовувати декілька моделей для класифікації, щоб об’єднати їх у один класифікатор. Цей метод, також дозволяє вирішити проблему перенавчання моделей та також неправильного визначення результатів, кожної з них.

Етапи виконання даного модуля бібліотеки, та особливості його використання:

- Перше і найважливіше – це визначення, яким чином відбувається отримання результатів передбачення комбінованого класифікатора після отримання прогнозів від кожної, окремо взятої моделі. Серед параметрів модулю є параметр voting з можливими значеннями: 'hard' і 'soft'.
    - У першому випадку підсумкова відповідь об'єднаного класифікатора буде відповідати «думці» більшості членів, які входять до нього. Наприклад, в об'єднаному класифікаторі використовується дані від трьох різних моделей. Дві з них на конкретному спостереженні пророкують відповідь «відгук позитивний», третій - «відгук негативний». Таким чином, для даного спостереження підсумковим пророкуванням буде «відгук позитивний», так як у нас 2 – «за» і 1 «проти».
    - У другому випадку, тобто при використанні значення 'soft' параметра voting йде повноцінне «голосування» і зважування пророкувань моделей для кожного класу, таким чином підсумковий відповідь об'єднаного класифікатор – це argmax суми передбачених ймовірностей. Для можливості використання такого методу «голосування» кожен класифікатор з вхідних в ваш ансамбль повинен

підтримувати метод predict_proba для отримання кількісної оцінки ймовірності входження в кожен з класів. Зверніть увагу, що не всі моделі класифікаторів підтримують цей метод і, відповідно, можуть бути використані в рамках VotingClassifier при використанні методу зважених ймовірностей (Soft Voting).

- Можливість одночасного використання модуля VotingClassifier і GridSearch для оптимізації гіперпараметров кожного з класифікаторів, що входять в ансамбль.

У статистиці добре відомо інтуїтивне міркування, згідно з яким усереднення результатів спостережень може дати більш стійку і надійну оцінку, оскільки послаблюється вплив випадкових флуктуацій в окремому вимірі. На аналогічній ідеї було засноване розвиток алгоритмів комбінування моделей, в результаті чого побудова їх ансамблів виявилося одним з найбільш потужних методів машинного навчання, нерідко перевершує за якістю прогнозів інші методи. Одним з рішень, що забезпечують необхідну різноманітність моделей, є їх повторне навчання на вибірках, випадково вибраних з генеральної сукупності, або інших подмножествах даних, сконструйованих з наявних (рис. 3.5). Для отримання стійкого прогнозу приватні передбачення цих моделей тим чи іншим чином комбінують, наприклад, за допомогою простого усереднення або голосування (можливо, зваженого).

У підрозділі 2.2 був описаний бутстреп – процедура генерації повторних випадкових вибірок з вихідного набору даних. Бутстреп-вибірки виробляються рівномірно, тому деякі вихідні параментри можуть бути відсутніми, а деякі – дублюватися: в середньому такі вибірки мають 60% унікальних даних.

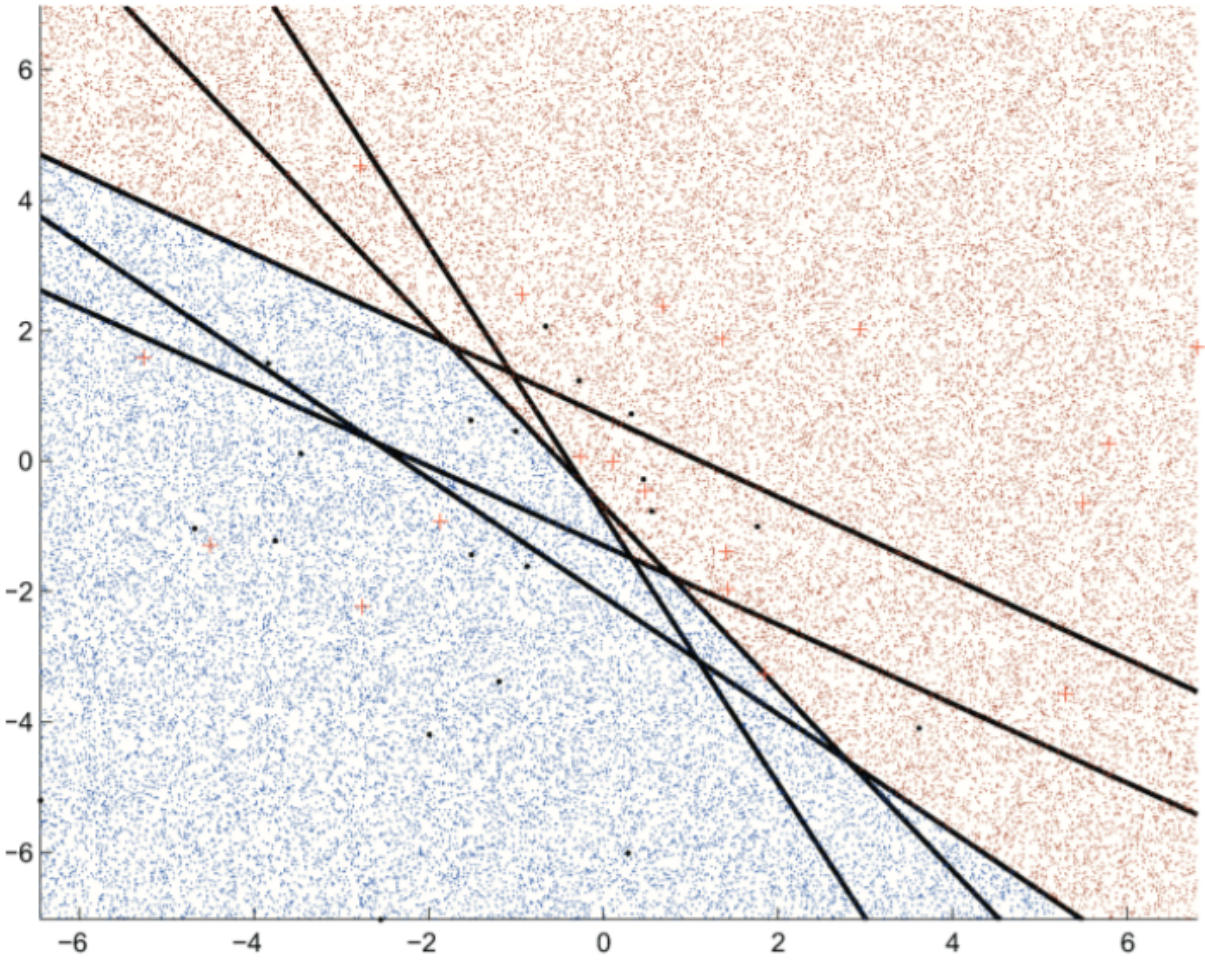


Рис. 3.5. Ансамбль з п'яти лінійних класифікаторів: кожен сегмент простору об'єктів відрізняється середніми можливостями передбачення класів

Бутстреп для побудови ансамблю моделей є корисним з використанням структур, чутливих до маленьких змін навчальних даних, деревоподібної форми. Описані в попередньому розділі дерева рішень зазвичай мають низьке зміщення, але страждають від високої дисперсії. Це означає, що якщо випадковим чином розіб'ємо навчальні дані на дві частини і побудуємо дерево рішень на основі кожної з них, то отримані результати можуть виявитися досить різними. Подібно до того, як урівнюються дані та пливають на дисперсію даних, так і зниження дисперсії прогнозу еквівалетне отриманою великої кількості порцій даних з генеральної сукупності, побудова предсказательной моделі по кожній навчальній

вибірці і усереднення отриманих прогнозів. Якщо замість окремих навчальних вибірок (яких нам, як правило, завжди не вистачає) виконати бутстреп і на основі згенерованих псевдо-вибірок побудувати B дерев регресії, то середній колективний прогноз буде мати більш низькою дисперсією. Ця процедура і називається беггінгом (скор. Від bootstrap aggregating). Як буде показано в наступних розділах, беггінг застосовувати не лише до дерев регресії, а й інших моделей: опорних векторів, лінійний дискримінант, байесовских ймовірностей і ін.

Метод випадкового лісу (Random Forest) являє собою подальше поліпшення беггінга дерев рішень, яке полягає в усуненні кореляції між деревами. Як і у випадку з беггінгом, будуємо кілька сотень дерев рішень по навчальним бутстреп-вибірках. Однак на кожній ітерації побудови дерева випадковим чином вибирається mm з pp підлягають розгляду предикторов і розбиття дозволяється виконувати тільки по одній з mm цих змінних. Сенс цієї процедури, яка опинилася досить ефективною для підвищення якості отримуваних рішень, полягає в тому, що з імовірністю p блокується який-небудь потенційно домінуючий предиктор, який прагне увійти в кожне дерево. Якщо домінування таких предикторів дозволити, то всі дерева в результаті будуть дуже схожі один на одного, а одержувані на їх основі передбачення будуть сильно корелювати і зниження дисперсії буде не настільки очевидним. Завдяки блокуванню домінантів, інші предиктори отримають свій шанс, і варіація дерев зростає. Вибір малого значення m при побудові випадкового лісу зазвичай буде корисним при наявності великого числа корелюють предикторов. Природно, якщо випадковий ліс будується з використанням $m = pm = p$, то вся процедура зводиться до простого беггінгу. Застосуємо методи беггінга і випадкового лісу до прогнозування даних по великій кількості водоростей в річках різного типу. Оскільки беггінг – це просто окремий випадок методу випадкового лісу, то можемо використовувати одну і ту ж функцію randomForest () пакета randomForest для R. Беггінг виконується, якщо задати параметр mtry = ncol (x).

Зауважимо, що бутстреп дає хорошу можливість провести спеціальну процедуру перехресної перевірки, звану тестом по «спостереженнями, які не потрапили в сумку» (out-of-bag observations). Оскільки ключова ідея беггінга складається в багаторазовому побудові моделей за спостереженнями з бутстреп-вибірок, то кожне конкретне дерево будується на основі приблизно двох третин усіх спостережень. Інша третина спостережень не використовується в навчанні, але цілком може бути використана для незалежного тестування: помилка на таких залишилися даних (out-of-bag error) є спроможною оцінкою помилки на контрольній вибірці Хоча набір отриманих в результаті беггінга дерев набагато складніше інтерпретувати, ніж окреме дерево, можна отримати цілих два узагальнених показника важливості кожного предиктора.

Іншим методом поліпшення прогнозів є бустінг (boosting), ідея якого полягає в итеративном процесі послідовного побудови приватних моделей. Кожна нова модель навчається з використанням інформації про помилки, зроблених на попередньому етапі, а результуюча функція являє собою лінійну комбінацію всього ансамблю моделей з урахуванням мінімізації будь штрафний функції. Подібно беггінгу, бустінг є загальним підходом, який можна застосовувати до багатьох статистичних методів регресії і класифікації. Тут ми обмежимося обговоренням градиентного бустінга в контексті дерев регресії. Бутстреп-вибірки в ході реалізації бустінга не створюються, але замість цього кожне дерево будується по набору даних X, rX, r, який на кожному кроці модифікується певним чином. На першій ітерації за значеннями вихідних предикторов будується дерево $f1\ (x)\ f1\ (x)$ і знаходиться вектор залишків $r1r1$. На наступному етапі нове регресійне дерево $f2\ (x)\ f2\ (x)$ будується вже не за навчальним даними XX, а по залишкам $r1r1$ попередньої моделі. Лінійна комбінація прогнозу по побудованим деревах дає нам нові залишки і цей ітераційний процес повторюється n раз. Завдяки побудові неглибоких дерев по залишкам, прогноз відгуку повільно поліпшується в областях, де одиночне дерево працює не дуже добре. Такі

дерева можуть бути досить невеликими, лише з кількома кінцевими вузлами. Параметр стиснення λλ регулює швидкість цього процесу, дозволяючи створювати комбінації дерев більш складної форми для "атаки" залишків.

Для побудови бустінг-моделей на основі дерев рішень можна використовувати функцію gbm () з пакету gbm (Generalized Boosted Models). Процес моделювання проходить під управлінням трьох гіперпараметрів:

- Число дерев *BB* (формальний параметр n.tree). На відміну від беггінга, бустінг може, хоча і повільно, приводити до перенавчання при надмірно великому *B*.
- Параметр стиснення λλ (shrinkage), який коригує величину вкладу кожного додаткового дерева і контролює швидкість, з якою відбувається навчання моделі при реалізації бустінга. Типові значення λλ варіюють від 0.01 до 0.001, і їх оптимальний вибір залежить від розв'язуваної проблеми. Для досягнення хорошої якості передбачень дуже низькі значення λλ вимагають дуже великого значення *BB*.
- Число внутрішніх вузлів *dd* (interaction.depth) в кожному дереві, яке контролює складність одержуваного в результаті бустінга ансамблю моделей. За своєю суттю, параметр *dd* відображає глибину взаємодій між предикторами в підсумковій моделі. Якщо ці взаємодії не дуже виражені, то добре працює *dd* = 1,1 і тоді додаткові дерева являють собою просто "пні" (stump), тобто містять тільки один внутрішній вузол. В такому випадку одержуваний в результаті бустінга ансамбль стає адитивної моделлю, оскільки кожен її член представлений тільки однієї змінної.

Тип розв'язуваної задачі регулюється параметром distribution, який визначає оптимізується функція:

- для вирішення завдань регресії задається значення "gaussian" – квадратичний штраф, або "laplace" – штраф за абсолютною величиною відхилення;
- для задач бінарної класифікації використовують значення "bernoulli" – функція крос-ентропії, або "adaboost" – експонентний штраф.

Використовуємо значення shrinkage = 0.001, встановлене функцією gbm () за замовчуванням. Функція summary () в відношення цього методу виводить список предикторов і відповідні їм значення показника важливості.

Виконаємо оптимізацію параметрів побудови градієнтного бустінга з використанням функції train (). Як сказано вище, таких параметрів три: Беручи до уваги, що параметри shrinkag і n.trees пов'язані обернено пропорційною залежністю, зменшимо число дерев до 50, одночасно збільшивши значення shrinkage в порівнянні з вживаними вище на рис. 3.6.

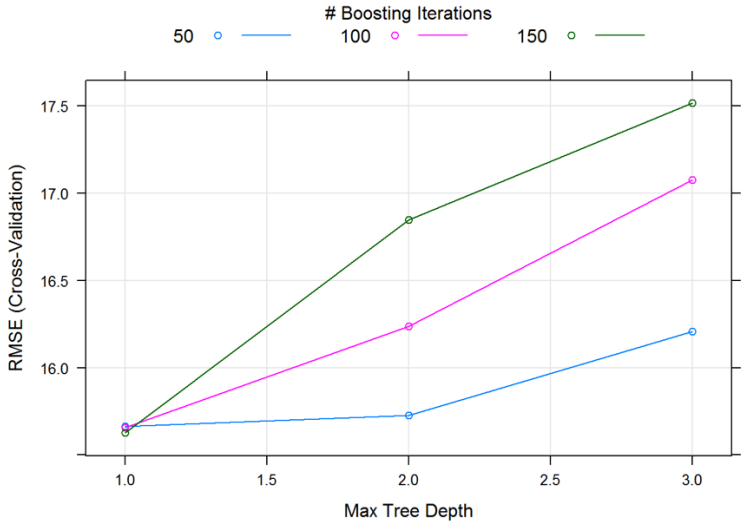


Рис. 3.6. Залежність помилки від числа дерев при бустингу (за результатами перехресної перевірки)

Надалі необхідно провести аналіз роботи програмної реалізації для автоматизованого тегування текстових даних.

3.3. Висновки до розділу 3

У даному розділі запропоновано засоби реалізації для кожного з етапів методу. Для більшості етапів вказано альтернативи, які можуть бути використані для реалізації методу сторонніми розробниками. Наведено огляд до організації програмного забезпечення. Для реалізації обрано організацію програмного забезпечення у вигляді комбінації моделей та передобробку даних завдяки методам «мішка слів», TF-IDF та градієнтний бустінг. Запропоновано структуру та наведено особливості реалізації кожного з алгоритмів для вирішення задачі реалізації автоматизованого тегування текстових описів для пошуку даних.

4. АНАЛІЗ РЕАЛІЗАЦІЇ СПОСОБУ ТА ОТРИМАНИХ РЕЗУЛЬТАТІВ

4.1. Тестування класифікатора голосування

Голосування – це один із найпростіших способів поєднання прогнозів із декількох алгоритмів машинного навчання. Класифікатор голосування – це не є власне класифікатор, а обгортка для набору різних моделей, які навчаються та оцінюються паралельно з метою використання різних особливостей кожного алгоритму. Кожну базову модель можна створити, використовуючи різні розщеплення одного навчального набору даних і того ж алгоритму, або використовуючи один і той же набір даних з різними алгоритмами. Для перевірки прогнозів для кожної моделі, збережемо їх у матриці, що називається передбаченнями, де кожен стовпець містить прогнози однієї моделі (рис. 4.1).

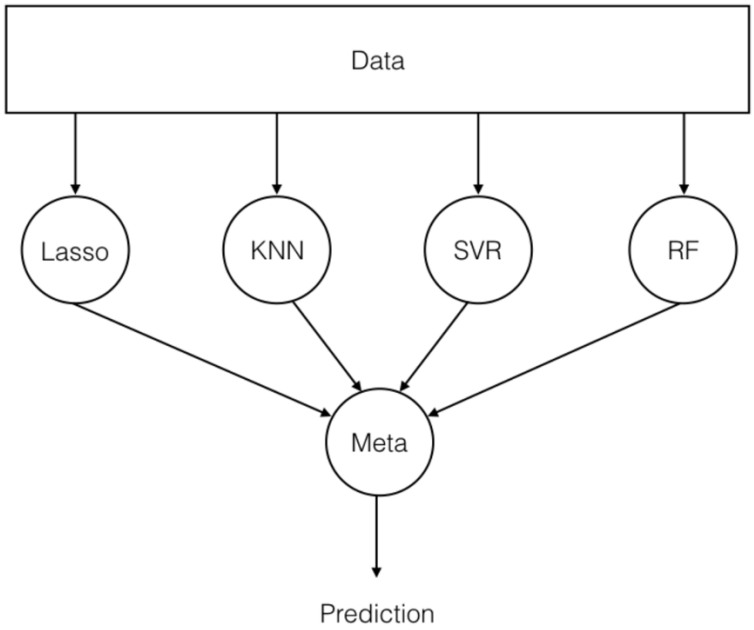


Рис. 4.1. Прогнозування методом голосування

Голосування повертає мітку класу як *argmax* суми передбачуваних ймовірностей. Конкретні ваги можуть бути призначені кожному класифікатору через параметр ваг. Коли передбачено зважування, прогнозовані ймовірності класу для кожного класифікатора збираються,

помножуються на вагу класифікатора та усереднюються. Потім кінцевий ярлик класу виводиться з мітки класу де була найвищою середня ймовірність. Середньозважені ймовірності для вибірки обраних моделей машинного навчаня відображаються на рис. 4.2.

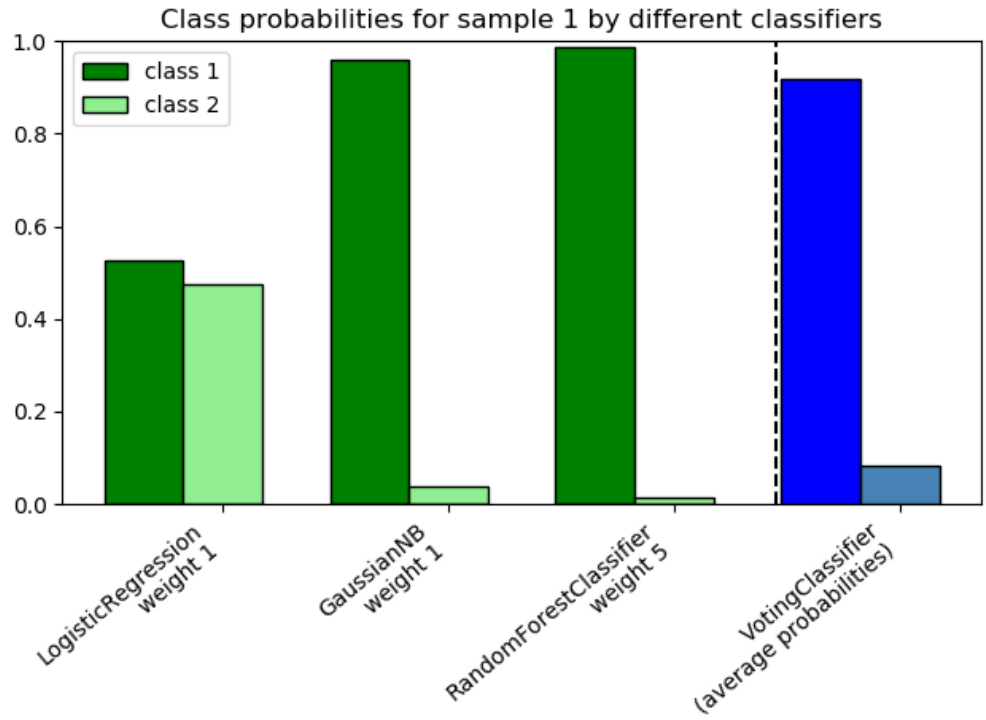


Рис. 4.2. Аналіз обраних моделей для навчання

За результатами тестування моделей описаних в розділі 3, отримано результат прогнозування, який є більшим ніж 90%, що вказує на правильний набір моделей для подальшого навчання ансамблю.

4.2. Тестування ансамблю моделей

Для перевірки доцільності роботи ансамблю моделей для автоматизованого тегування текстових даних було обрано сайт indeed.com для збору даних. Сайт пропонує найбільшу в світі пошукову систему. Теги знайдені на сайті дозволяють користувачам фільтрувати та легше знаходити роботу.

Наприклад, враховуючи повну характеристику роботи, програма повинна точно витягувати інформацію, наприклад, кількість необхідних років досвіду.

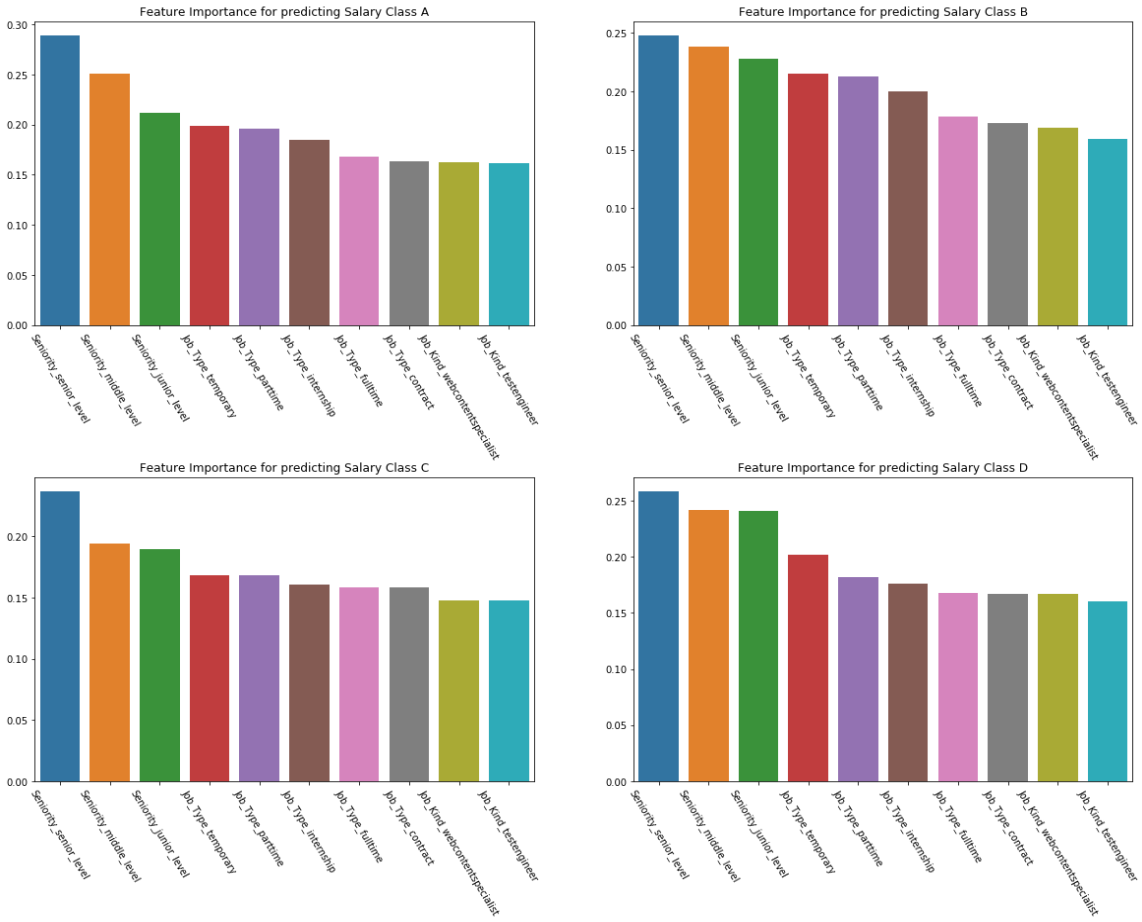


Рис. 4.3. Діаграма визначення важливості міток серед векторизованого набору даних

Незважаючи на те, що, можливо, розмір міток не допомагає у зменшенні похибки, можна зробити висновок із цих діаграм на рис. 4.3, що такі характеристики нашого набору даних є релевантними при визначенні категорії зарплат:

- стаж – досвід роботи впливає дуже сильно на всі категорії, будучи першим за важливістю коефіцієнтом за абсолютною величиною;
- тип роботи (Job_Type);
- локація чи координати компанії;

- і нарешті, у є всі категорії заробітної плати, два види робочих місць – це заголовки, які насправді містять іншу назву категорій: спеціаліст із веб-контенту та інженер-тестувальник.

Цей набір даних містить сотні параметрів, але ми бачимо чітко виражений набір властивостей, які важливі для визначення категорій, до яких можна віднести проаналізований опис вакансії.

На жаль, дана модель (логістична регресія) не надала бажаних результатів, тому було вирішено використати такі моделі:

- a KNN model with bagging;
- a Decision Tree model with boosting.

Кожна окремо взята модель під час навчання мала похибку більше 35%, що є недопустимим значенням, однак, програмне рішення створене за допомогою комбінації моделей, зазначених вище, отримало кращу точність визначення категорій вакансій, ніж кожна окрема взята з цих моделей. Середня точність знаходження основних категорій вакансій, визначених комбінацією вищезгаданих моделей (або ансамблю моделей) є 86%, що означає, що принаймні більше 80% усіх прогнозованих значень для кожної категорії було правильним.

Зрештою, базовий набір даних отримав під час тренування досить надійний ансамбль моделей, так как я жодна вибірка з категорій не вплинула на відхилення при різній величині даних. На наступних матрицях на рис. 4.4 бачимо, що тренувальні і тестові групи випробувань прогнозувались із високою точністю для чотирьох різних категорій зарплат та для різних розмірів даних, що вказує на правильність роботи алгоритму.

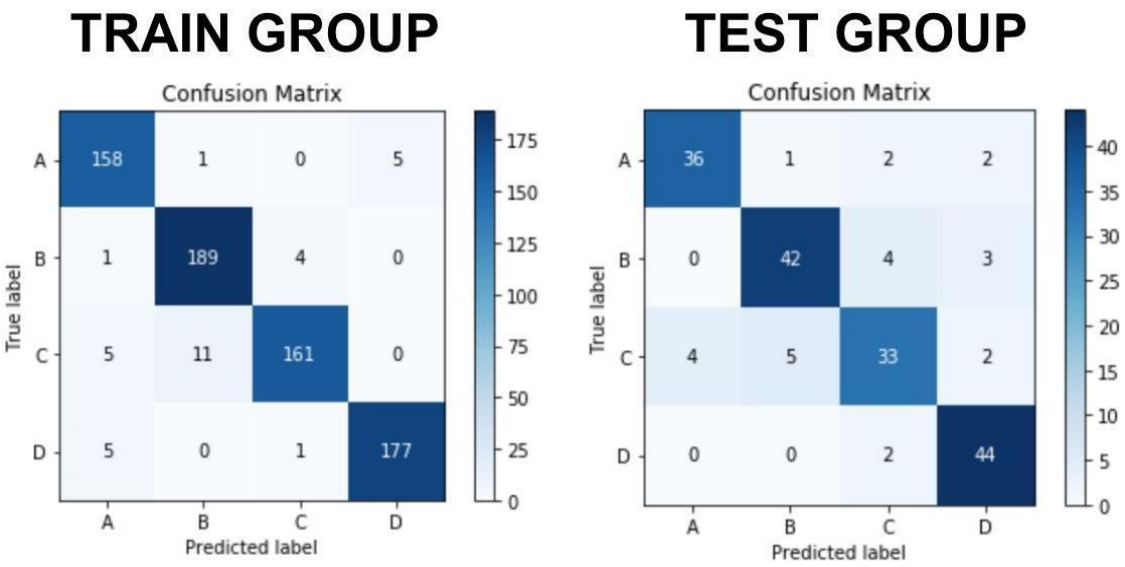


Рис. 4.4. Матриця відхилень спрогнозованих категорій

4.3. Висновки до четвертого розділу

В даному розділі наведено результати аналізу доцільності використання методу. Практичні результати підтвердили теоретичні гіпотези про те, що використання ансамблю моделей машинного навчання надає виграш для задач, коли текстові данні залежать від часу та інших прихованих додаткових властивостей які можна знайти у тексті. Також отримано підтвердження того, що грамотне використання методів обробки даних, запропоноване програмною реалізацією, збільшує точність автоматизованого тегування текстових даних.

5. ПОБУДОВА БІЗНЕС-МОДЕЛІ

Розглянемо рішення автоматизованого тегування текстових документів конкретно на прикладі описів вакансій, оскільки вони наглядно демонструють вирішення проблем для ринку підбору IT-персоналу за запитами користувачів.

5.1. Огляд проблеми

Створення програмного забезпечення для підбору IT-компаній за описом користувача потребує чітко визначеного моделювання всіх можливих варіантів вимог та сценаріїв для забезпечення найточнішого результату. Задача автокатегоризації компаній за описом потребує багато знань у галузі комп'ютерної лінгвістики, тому ймовірініть допущення помилок є великою. Автокатегоризація текстових документів відноситься до галузі комп'ютерної лінгвістики, що активно розвивається. Це пов'язано зі збільшенням об'ємів текстових даних у мережі і їх просто неможливо обробити вручну.

По-перше, високо кваліфікованим спеціалістам, які шукають роботу, доводиться самостійно знаходити додаткову інформацію на сайтах компаній, як можливо їм підійдуть, але на кожну таку вакансію доводиться витрачати від 10 до 60 хвилин часу, що не є доцільним, якщо в більшості випадків розробник розглядає декілька вакансій.

По-друге, також велика кількість вакансій які надходять у повідомленнях(на пошту, linkedin), можуть не відповідати запитам розробників по стеку технологій, типу компанії або категоріям проекту.

По-третє, існує велика кількість компаній, які необхідно відфільтрувати з пошуку через велику кількість негативних відгуків від самих розробників, які працювали там раніше чи проходили співбесіду.

Всі вище описані пункти представлені у схемі «Дерева проблем», зображеної на рис. 5.1.

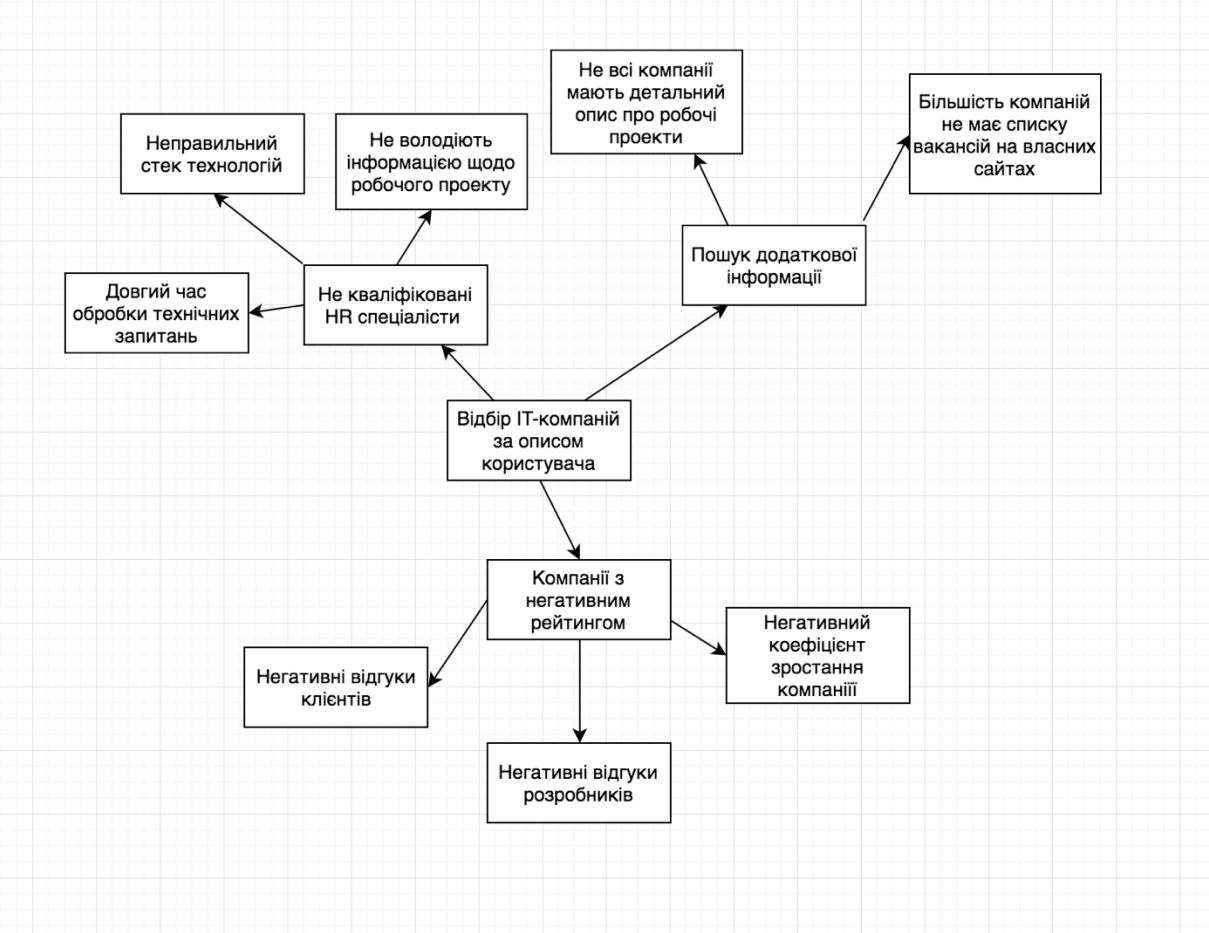


Рис. 5.1. Дерево проблем

5.2. Зацікавлені сторони

Для вирішення даної проблеми, існує кілька зацікавлених сторін.

Основною зацікавленою стороною є розробники. Вони витрачають щорічно витрачають чимало часу на пошук відповідної їм роботи за стеком та софт складовою в компанії. Завдяки їм, IT-компанії виконують замовленяя клієнтів та отримують прибуток.

Також зацікавленими сторонами є IT-компанії. Саме їх пости з вакансіями стають об’єктами для аналізу данних, та можливістю для фахівців обрати для себе нову роботу чи новий стек технологій.

Також агенства по підбору персоналу завдяки данному сервісу зможуть краще зрозуміти потреби кандидатів та надсилати вже відфільтровану інформацію у повідомлення на пошту чи linkedin.

У табл. 5.1 продемонстровані усі групи зацікавлених сторін, їх інтереси, та вплив (міра зацікавленості у вирішенні наявних проблем).

Таблиця 5.1

Міра зацікавленості у вирішенні наявних проблем

Зацікавлена сторона	Інтерес зацікавленої особи	Вплив зацікавленої особи	Стратегії приваблення зацікавлених сторін
Розробники	Економній та надійній спосіб знаходження списку вакансій та компаній	Високий	Проведення презентації для представників зацікавлених осіб
IT-компанії	Зніження ризику відхилення оферу під час процесу найму	Високий	Участь у спеціалізованих виставках та форумах
Агенство з підбору персоналу	Фільтрація та збір необхідної інформації для кандидатів	Низький	Реклама в соціальних мережах

5.3. Комерційне рішення. Основні характеристики

Відповідно до вище зазначених проблем, можна описати кінцевий продукт, що має їх вирішувати. Даний програмний продукт буде реалізовувати описану у попередній розділах автокатегоризацію тексту, що базується на описах користувачів. Даний метод дозволяє ефективно визначати список IT-компаній, які зацікавлять розробників. Дані зміни будуть помітними під час пошуку для кожного розробника та IT-компаній, адже кожна із сторін буде мотивована закрити вакансію із ідейно зацікавленими розробниками.

По-перше, зменшиться кількість витрат на служби та агенста пошуку кандидатів, адже самі кандидати будуть самі зацікавлені в отриманні роботи в певній компанії. По-друге, IT-компанії та розробники будуть витрачати менше часу на процес найму та прийняття оферу.

Саме тому, очевидно, що клієнтом даного продукту є IT-компанії, а співпраця буде побудована на моделі співробітництва бізнесу для бізнесу, або як він ще називається B2B.

Даний програмний продукт повинен бути зрозумілим у використанні та надавати всю необхідну інформацію для розробників, щоб перед подачею резюме не виникало низки питань, шодо проекту чи набору технологій. Зважаючи на вище сказане, можна зробити висновок, що проблем з розробкою системи виникати не повинно, або вони повинні швидко і легко вирішитися за допомогою невеликої кількості спеціалістів.

5.4. Конкурентні переваги рішення

Так як вже зазначалося, що дана сфера є новою, то якісних реалізацій сервісів по підбору IT-компаній за описом користувача – не існує. Зараз існують окремі рішення, але вони існують на ринку США та Індії. Тому єдиним конкурентом залишаються служби по розсилці вакансій по завантаженому резюме.

Основний конкурент має безліч недоліків, які були описані в підрозділі 5.1. Майже всі недоліки, що має аналог, вирішуються за сервісу по підбору IT-компаній за описом користувача.

Отже, конкурентними перевагами програмного продукту, що пропонується, є:

- зменшення кількості витраченого часу на обробку інформації по вакансії;
- збільшення коефіцієнту найму для айті компаній;
- покращення якості наданої інформації для розробників;

- унікальний метод автокатегоризації опису користувача за існуючим списком данних.

5.5. Клієнти. Сегменти ринку споживання

Для досягнення максимальної ефективності діяльності команди проекту рекомендується провести сегментацію ринку клієнтів, тобто виділити більш-менш однорідні групи користувачів, зацікавлених в пропонованому продукті.

Таким чином, сегментацію ринку клієнтів для конструктора схем бізнес-процесів можна провести за двома категоріями:

- обсяг запланованих наймів;
- можливість отримати список компаній за отриманими категоріями опису.

Згідно з цими двома критеріями, потенційних клієнтів можна розділити на 2 групи:

- розробники, які готові заплатити гроші за збережений час під час підбору компаній
- IT-компанії, які хочуть підвищити якість найму персоналу;

5.6. Унікальна ціннісна пропозиція

Ціннісна пропозиція – це пояснення того, як продукт вирішує проблему. Його можна скласти за формулою: Ціннісне пропозицію = Проблема + Рішення / Продукт.

У дереві проблем було виділено низку проблем, а у зацікавлених сторонах – було визначено очікування відповідних сторін від продукту.

IT-компанії бажають отримати:

- Відбір кандидатів – ШІ розсилає оголошення про пошук співробітників на відповідні ресурси – дошки оголошень про роботу. Коли кандидат відповідає на вакансію, його резюме отримує система ШІ, при відповідності вимог здобувача

запрошують на співбесіду. Незалежно від того, чи займається штучний інтелект аналізом резюме, ідентифікацією ключових слів або збором даних для подальшого сортування, в кінцевому підсумку AI формує стек цифрових резюме.

- Ефективну комунікацію – компанії зацікавлені у тому, щоб бренд був впізнаваним і викликав позитивні асоціації у всіх без винятку претендентів. Оптимізація процесу подання резюме – один з ефективних способів привернути до себе кандидата. Завдяки ШІ, а саме функції чат-ботів або моментального обміну повідомленнями, заповнити анкету можна всього за кілька хвилин, без дзвінків і обміну електронними листами.
- Оптимізація рекрутингу – завдяки платформам і сервісам на базі ШІ рекрутер отримує можливість займатися тим, чого не може зробити комп'ютер: проводить співбесіди і відбирає персонал, ідеально відповідає вимогам компанії, приймає рішення, ґрунтуючись на загальному враженні від кандидата.

  А для розробників значно зберегти час.

Дійсно, запропоноване рішення, дозволяє частково задовольнити всі з наведених вище вимог зацікавлених сторін та вирішити наведені нижче проблем

Отже, основною унікальною ціннісною пропозицією є розроблений сервіс по підбору IT-компаній за описом користувача, що враховує зв'язки між користувачами.

5.7. Доходи та витрати

Оцінюючи витрати на проект, варто виділити дві основні категорії: стартові і щомісячні витрати. Стартові витрати – це гроші, які вкладаються один раз на початку проекту, а щомісячні, відповідно, вкладаються в розвиток проекту щомісяця.

Таким чином, в даному проекті до стартових відносяться витрати:

- на юридичні послуги;
- на ріелтерські послуги, мінімальні меблі, канцелярію;
- на комп'ютери та комплектуючі, ПЗ;
- на транспортні витрати.

Щомісячні витрати на проект поділяються на наступні групи:

- зарплати за посадами;
- оренда офісного приміщення;
- комунікації;
- податки;
- непередбачені витрати

Загальні витрати на проект наведено в табл. 5.2.

Таблиця 5.2

Загальні витрати

Вид витрат	Вартість ($)
Стартові	
Юридичні послуги	500
Рієлтерські послуги, мінімальні меблі, канцелярія	3100
Комп’ютери та комплектуючі, програмне забезпечення	3000 * 6
Транспортні витрати	100
Щомісячні	
Зарплати за посадами	2500 * 6
Оренда офісного приміщення	400
Комунікації	200
Податки	990
Непередбачені витрати	1000

Таким чином, підсумувавши сказане, можна сказати, що сума, необхідна для старту проекту становить 21700$, а щомісячні витрати складуть 17590$.

Узагальнено кажучи, можна виділити дві стратегії отримання прибутку в сервісах такого формату: від розробників і від IT-компаній. Для того щоб заробити якомога більшу кількість грошей на продукті, необхідно об'єднати обидві стратегії. У випадку з сервісом по вибору IT-компаній це можна зробити створивши кілька типів підписок для використання програмного забезпечення для обох типів користувачів та завдяки рекламі. Підписка, яка призначається для використання компаніями та її співробітниками, буде надавати функціональність системи по розсилці вакансій зацікавлених розробників та данні цих користувачів. Вартість ліцензії такого типу буде розподілена на тимчасові проміжки використання. Вартість одного часового проміжку використання продукту становитиме 50$. Підписка, яка призначається для використання розробниками, буде надавати можливість користувачам отримати списки релевантних компаній за категоріями з опису. Поширюватися така ліцензія буде безкоштовно, але отримувати дохід на її використанні ми будемо за рахунок переглядається реклами. Необхідно, також, ввести тимчасову підписку (trial), яка не обмежена за функціональністю, але обмежена за часом використання. Призначатися цей тип ліцензії буде для ознайомлення з можливостями продукту.

5.8. Бізнес-модель

Узагальнимо, все написане вище у лаконічну бізнес-модель у вигляді lean canvas.

Споживачі: IT-компанії, розробники та hr-агенства.

Проблема: витрата часу розробників; не кваліфіковані HR спеціалісти; компанії з негативним рейтингом; відсутність додаткової інвофрмації.

Рішення: сервіс, по автокатегоризації тексту, що базується на описах користувачів для підбору списку IT-компаній.

Унікальна ціннісна пропозиція: розроблений сервіс по підбору IT-компаній за описом користувача.

Потоки доходів: доходи від продажу підписок; доходи від підтримки програмного забезпечення; реклама;

Структура витрат: утримання персоналу для надання технічної підтримки (виплати заробітних плат, соціальних виплат); утримання робочих місць для персоналу (оплата за оренду офісу та комунальні послуги); податкові витрати; оплата послуг юриста, бухгалтера, прибиральниці.

Також в канву бізнес-моделі включаються структурні блоки: прихована перевага (перевага, яку не можливо скопіювати або купити), ключові метрики (основні показники, що вимірюються) та канали (шляхи до користувачів).

Канали: через відділи співпраці чи інтеграції відповідних соціальних мереж.

Ключові метрики: кількість проданих підписок.

Прихована перевага: врахування специфіки сервісів по підбору персоналу.

Бізнес-модель наведена у зведеному вигляді у табл. 5.3.

Таблиця 5.3

<table>
<tr><td colspan="5">Канва бізнес-моделі</td></tr>
<tr><td>Проблема</td><td>Рішення</td><td>Унікальна ціннісна пропозиція</td><td>Прихована перевага</td><td>Споживачі</td></tr>
<tr><td>Витрата часу розробників</td><td rowspan="4">Програмне забезпечення для автоматизованого тегування текстових документів</td><td rowspan="4">Розробка ПЗ для тегування текстових документів і зменшення витрат часу та коштів</td><td rowspan="4">Врахування специфіки сервісів для тегування текстових документів</td><td rowspan="4">Компанія, розробник та HR</td></tr>
<tr><td>Не кваліфіковані HR спеціалісти</td></tr>
<tr><td>Компанії з негативним рейтингом</td></tr>
<tr><td>Відсутність додаткової інформації</td></tr>
<tr><td colspan="2">Структура витрат</td><td colspan="3">Потоки доходів</td></tr>
<tr><td colspan="2">Утримання персоналу для надання технічної підтримки</td><td colspan="3" rowspan="3">Доходи від продажу програмної ліцензії та підтримки програмного забезпечення</td></tr>
<tr><td colspan="2">Утримання робочих місць для персоналу</td></tr>
<tr><td colspan="2">Податкові</td></tr>
</table>

Отже, зважаючи на дані у табл. 5.3, можна зробити висновок, що запропонований проект, який реалізує описаний у дисертацій спосіб автокатегоризації тексту, має перспективи у своїй подальшій реалізації. Звичайно, проведений аналіз не враховує всіх ризиків та факторів, таких як специфіка оподаткування у країні ведення бізнесу, проте навіть наявних досліджень достатньо, щоб прогнозувати комерційний успіх продукту та його окупаємість.

5.9. Висновки до розділу 5

У даному розділі було проведено аналіз поточної ситуації у сфері автокатегоризації тексту по опису, виявлено наявні проблеми та підсумовано у відповідному дереві проблем. Наряду з проблемами було виділено основні зацікавлені сторони у вирішенні існуючих недоліків, ступінь впливу даних сторін на вирішення проблем. Як наслідок було запропоновано комерційне рішення з конкурентними перевагами, що задовольняє інтереси зацікавлених осіб, та виділено унікальну ціннісну пропозицію запропонованого продукту. Було проведено аналіз майбутніх клієнтів, досліджено сегменти ринку споживання. Це дозволило спрогнозувати потенційні доходи та витрати на реалізацію продукту. У результаті була описана бізнес-модель, що обґрунтовує доцільність реалізації даного продукту та прогнозує його потенційну окупаємість та прибутковість в подальшому.зацікавлені сторони у вирішенні існуючих недоліків, ступінь впливу даних сторін на вирішення проблем. Як наслідок було запропоновано комерційне рішення з конкурентними перевагами, що задовольняє інтереси зацікавлених осіб, та виділено унікальну ціннісну пропозицію запропонованого продукту. Було проведено аналіз майбутніх клієнтів, досліджено сегменти ринку споживання. Це дозволило спрогнозувати потенційні доходи та витрати на реалізацію продукту. У результаті була описана бізнес-модель, що обґрунтовує доцільність реалізації даного продукту та прогнозує його потенційну окупаємість та прибутковість в подальшому.

ВИСНОВКИ

В даній роботі запропоновано нове вирішення задачі автоматизованого тегування текстових документів, яке знайшло відображення в розробці ансамблю моделей класифікації текстових описів.

1. Оцінено сучасний стан проблеми, обґрунтовано актуальність обраного напрямку досліджень, визначені мета і задачі дослідження.
2. Розглянуто основні методи класифікації текстових даних, які можуть бути використані для автоматизованої категоризації текстових описів; наведені теоретичні засади щодо побудови моделі машинного навчання та класифікаторів текстів.
3. Проаналізовано критерії ефективності існуючих методів класифікації текстових даних; досліджено методи оптимізації, які можуть бути використані для побудови ансамблю моделей та класифікаторів машинного навчання для текстової обробки; розглянуто методи для аналізу великих об’ємів даних для виділення категорій у тексті.
4. Сформульовано основні етапи розроблення ансамблю моделей і обраних класифікаторів (збір даних, обробка, підрахунок коефіцієнтів для кожного алгоритму, аналізатор рішень) для розв’язання задач автоматизованого тегування текстових документів.
5. Запропоновано алгоритм побудови ансамблю моделей; досліджено проблеми, пов’язані з аналізом і класифікацією текстових даних.
6. Запропоновано програмну реалізацію комбінації моделей машинного навчання для обробки текстових даних.
7. Проаналізовані результати роботи ансамблю моделей машинного навчання для класифікації текстових документів та виявлено, що комбінація класичних класифікаторів, дозволяє отримати набагато

ефективніші результати для текстів з великою кількістю категорій за малий проміжок часу ніж інші методи, використані для порівняння.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Наївний баєсів класифікатор [Електронний ресурс] . – Режим доступу: https://uk.wikipedia.org/wiki/Наївний_баєсів_класифікатор. – 01.04.2014.
2. Правильность и ошибочность [Електронний ресурс]. — Режим доступу: https://studbooks.net/2259240/informatika/pravilnost_oshibochnost — (16.03.2015) — Назва з екрана.
3. Метод k-найближчих сусідів [Електронний ресурс]. — Режим доступу : https://replace.org.ua/topic/8112/— (12.03.2015) — Назва з екрана.
4. Дерево ухвалення рішень [Електронний ресурс]. — Режим доступу : https://uk.wikipedia.org/wiki/Дерево ухвалення рішень — (10.03.2015) — Назва з екрана.
5. Categorizing and Tagging Words [Електронний ресурс]. — Режим доступу : https://www.nltk.org/book/ch05.html— (10.03.2015) — Назва з екрана.
6. Comparasion dashboard [Електронний ресурс]. — Режим доступу : https://www.quora.com/Are-there-any-algorithms-for-short-text-classification — (10.03.2015) — Назва з екрана.
7. Datawarehousing with PostgreSQL [Електронний ресурс]. — Режим доступу : https://wiki.postgresql.org/images/3/38/PGDay2009-EN-Datawarehousing_with_PostgreSQL.pdf — (10.03.2015) — Назва з екрана.
8. Text Classification [Електронний ресурс]. — Режим доступу : https://monkeylearn.com/text-classification/ — (17.04.2016) — Назва з екрана.
9. Tagging raw job descriptions [Електронний ресурс]. — Режим доступу : https://www.hackerrank.com/contests/indeed-ml-codesprint-2017/challenges/tagging-raw-job-descriptions — (10.03.2015) — Назва з екрана.
10. PgBouncer [Електронний ресурс]. — Режим доступу : http://evtuhovich.ru/blog/2012/02/12/pgbouncer/ — (10.03.2015) — Назва з екрана.

11. Using EXPLAIN [Електронний ресурс]. — Режим доступу : https://wiki.postgresql.org/wiki/Using_EXPLAIN — (10.03.2015) — Назва з екрана.
12. Explaining EXPLAIN [Електронний ресурс]. — Режим доступу : https://wiki.postgresql.org/images/4/45/Explaining_EXPLAIN.pdf — (10.03.2015) — Назва з екрана.
13. Caching [Електронний ресурс]. — Режим доступу : http://www.fullstackpython.com/caching.html — (10.03.2015) — Назва з екрана.
14. LRU, метод вытеснения из кэша [Електронний ресурс]. — Режим доступу : http://habrahabr.ru/post/136758/ — (10.03.2015) — Назва з екрана.
15. Why do we need a message brokers like rabbitmq over database like postgres? [Електронний ресурс]. — Режим доступу : http://stackoverflow.com/questions/13005410/why-do-we-need-a-message-brokers-like-rabbitmq-over-database-like-postgres — (10.03.2015) — Назва з екрана.
16. TWP-partitioning [Електронний ресурс]. — Режим доступу : http://www.oracle.com/technetwork/database/options/partitioning/twp-partitioning-11gr2-2009-09-130569.pdf — (10.03.2015) — Назва з екрана.
17. JS vs Python vs GO [Електронний ресурс]. — Режим доступу : https://hackernoon.com/javascript–vs–python–vs–go–the–deal–breaker–e54328ffd550 — (10.03.2015) — Назва з екрана.
18. Documentation [Електронний ресурс]. — Режим доступу : http://flywaydb.org/documentation/ — (10.03.2015) — Назва з екрана.
19. Moving from mysql to postgresql, best features I was missing? [Електронний ресурс]. — Режим доступу : http://stackoverflow.com/a/5023936/3070670 — (10.03.2015) — Назва з екрана.
20. Reek [Електронний ресурс]. – Режим доступу : https://github.com/nodegit/nodegit. – (04.07.2015) – Назва з екрана.

21. Go–git [Електронний ресурс]. – Режим доступу : https://github.com/src–d/go–git. – (06.05.2017) – Назва з екрана.
22. Clustering [Електронний ресурс]. — Режим доступу : http://scikit-learn.org/stable/modules/clustering.html — (10.03.2015) — Назва з екрана.
23. Profile, cProfile, and pstats – Performance analysis of Python programs [Електронний ресурс]. — Режим доступу : http://scikit-http://pymotw.com/2/profile/ — (10.03.2015) — Назва з екрана.
24. Flay [Електронний ресурс]. — Режим доступу : https://github.com/seattlerb/flay — (10.03.2015) — Назва з екрана.
25. Flog [Електронний ресурс]. — Режим доступу: https://github.com/seattlerb/flog — (10.03.2015) — Назва з екрана.
26. Choosing framework [Електронний ресурс]. — Режим доступу : https://smashingboxes.com/blog/choosing–a–front–end–framework–angular–ember–react/ — (10.03.2015) — Назва з екрана.

ДОДАТКИ

Додаток 1

Лістинг програми

Лістинг 1. base.py

```
import sys
import random
from time import sleep
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.decomposition import TruncatedSVD
from sklearn.neural_network import MLPClassifier

def stopw(w):
    stopwords = ['&', '\uf02d', 'all', 'just', 'being', 'over', 'both',
'through', 'yourselves', 'its', 'before', 'herself', 'had', 'should',
'to', 'only', 'under', 'ours', 'has', 'do', 'them', 'his', 'very',
'they', 'not', 'during', 'now', 'him', 'nor', 'did', 'this', 'she',
'each', 'further', 'where', 'few', 'because', 'doing', 'some', 'are',
'our', 'ourselves', 'out', 'what', 'for', 'while', 'does', 'above',
'between', 't', 'be', 'we', 'who', 'were', 'here', 'hers', 'by', 'on',
'about', 'of', 'against', 's', 'or', 'own', 'into', 'yourself', 'down',
'your', 'from', 'her', 'their', 'there', 'been', 'whom', 'too',
'themselves', 'was', 'until', 'more', 'himself', 'that', 'but', 'don',
'with', 'than', 'those', 'he', 'me', 'myself', 'these', 'up', 'will',
'below', 'can', 'theirs', 'my', 'and', 'then', 'is', 'am', 'it', 'an',
'as', 'itself', 'at', 'have', 'in', 'any', 'if', 'again', 'no', 'when',
'same', 'how', 'other', 'which', 'you', 'after', 'most', 'such', 'why',
'a', 'off', 'i', 'yours', 'so', 'the', 'having', 'once']
    return w in stopwords

def create_labels(my_array, str):
    labels = []
    for i in range(len(my_array)):
        my_l = 0
        for l in my_array[i]:
            if str == l:
                my_l = 1
        labels.append(my_l)
    return labels

def work(test_data, labels):
    my_vector = CountVectorizer(input='content',ngram_range=(1,2))
    X_train_counts = my_vector.fit_transform(train,)
    tf_transformer = TfidfTransformer(use_idf=True,).fit(X_train_counts)
    X_train_tf = tf_transformer.transform(X_train_counts)
    svd = TruncatedSVD(n_components=50, random_state=9)
    X_train = svd.fit_transform(X_train_tf)

    clf3 =  MLPClassifier(max_iter = 400).fit(X_train, labels)

    X_new_counts = my_vector.transform(test_data)
    X_new_tfidf = tf_transformer.transform(X_new_counts)
    X_new = svd.transform(X_new_tfidf)
    return clf3.predict(X_new)
```

Лістинг 2. train.py

```
file = open("train.tsv", "rb")
raw_data = file.read().decode()
file.close()

docs = raw_data.split("\n")
docs2 = docs[1: ]

random.seed(0)
random.shuffle(docs2)

train = []
labels = []

for d in docs2:
    d = d.lower()
    d = d.split("\t")
    if len(d) != 0:
        labels.append(d[0].split())
        text = d[1].split()
        train.append(' '.join([i for i in text if not stopw(i)]))

label_1_year = create_labels(labels, '1-year-experience-needed')
label_2_year = create_labels(labels, '2-4-years-experience-needed')
label_5_year = create_labels(labels, '5-plus-years-experience-needed')
label_AS = create_labels(labels, 'associate-needed')
label_BS = create_labels(labels, 'bs-degree-needed')
label_full_time = create_labels(labels, 'full-time-job')
label_hourly = create_labels(labels, 'hourly-wage')
label_1icence = create_labels(labels, 'licence-needed')
label_MS = create_labels(labels, 'ms-or-phd-needed')
label_part_time = create_labels(labels, 'part-time-job')
label_salary = create_labels(labels, 'salary')
label_supervising = create_labels(labels, 'supervising-job')
```

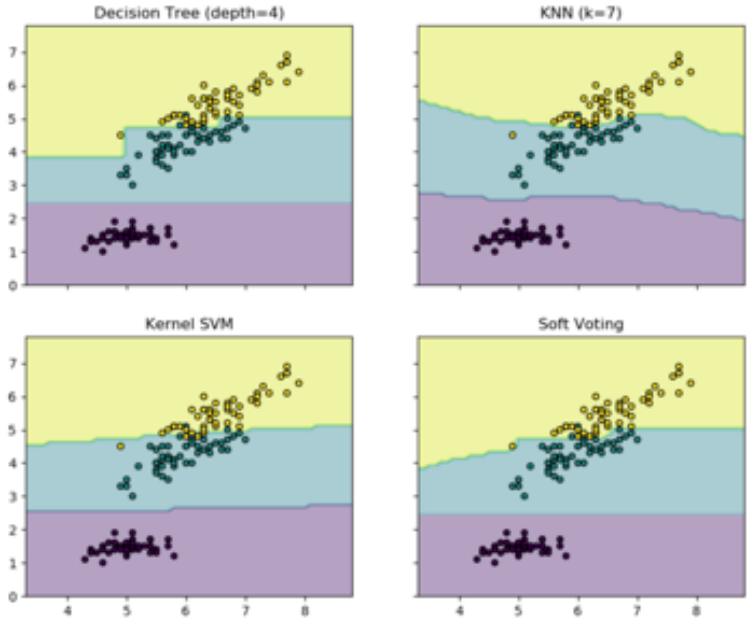
Лістинг 3. Test.py

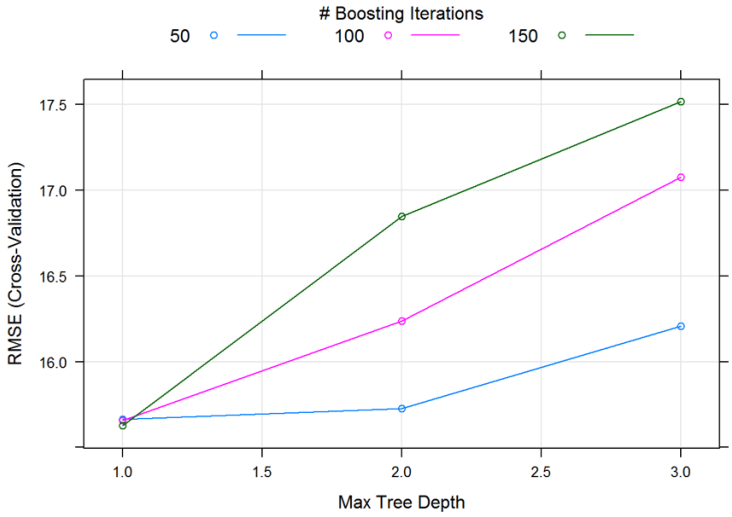
```
pr0 = work4(test, label_1_year)
for i in range(len(test)):
    if pr0[i] == 1:
        sleep(0.1)
        print(i, '1-year-experience-needed')
        result[i] +=' 1-year-experience-needed'
print("labels1 are done!")

pr1 = work4(test,label_2_year)
for i in range(len(test)):
    if pr1[i] == 1:
        sleep(0.1)
        print(i, '2-4-years-experience-needed')
        result[i] += ' 2-4-years-experience-needed'
print("labels2 are done!")

pr2 = work4(test, label_5_year)
for i in range(len(test)):
    if pr2[i] == 1:
        sleep(0.1)
        print(i, '5-plus-years-experience-needed')
        result[i] += ' 5-plus-years-experience-needed'
print("labels3 are done!")

pr3 = work4(test, label_AS)
for i in range(len(test)):
    if pr3[i] == 1:
        sleep(0.1)
        print(i, 'associate-needed')
        result[i] += ' associate-needed'
print("labels4 are done!")

pr4 = work4(test, label_BS)
for i in range(len(test)):
    if pr4[i] == 1:
        sleep(0.1)
        print(i, 'bs-degree-needed')
        result[i] += ' bs-degree-needed'
print("labels5 are done!")

pr5 = work4(test, label_full_time)
for i in range(len(test)):
    if pr5[i] == 1:
        sleep(0.1)
        print(i, 'full-time-job')
        result[i] += ' full-time-job'
print("labels6 are done!")

pr6 = work4(test, label_hourly)
for i in range(len(test)):
    if pr6[i] == 1:
        sleep(0.1)
        print(i, 'hourly-wage')
        result[i] +=' hourly-wage'
print("labels7 are done!")

pr7 = work4(test, label_1icence)
for i in range(len(test)):
```

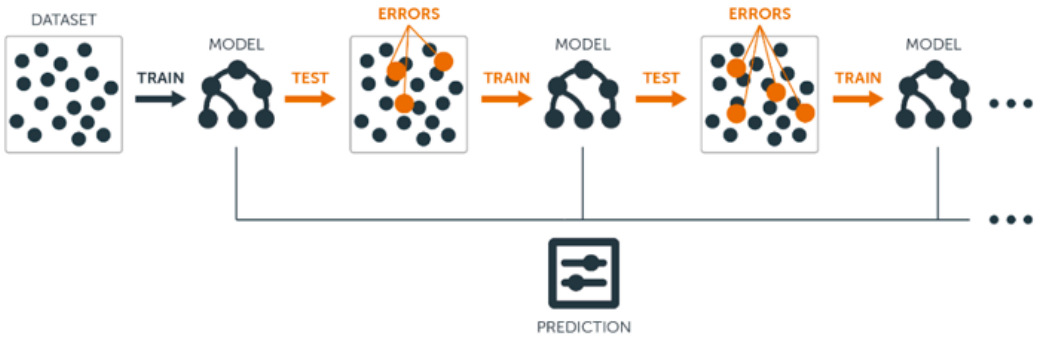
```
    if pr7[i] == 1:
        sleep(0.1)
        print(i, 'licence-needed')
        result[i] +=' licence-needed'

print("labels8 are done!")

pr8 = work4(test, label_MS)
for i in range(len(test)):
    if pr8[i] == 1:
        sleep(0.1)
        print(i, 'ms-or-phd-needed')
        result[i] +=' ms-or-phd-needed'
print("labels9 are done!")

pr9 = work4(test, label_part_time)
for i in range(len(test)):
    if pr9[i] == 1:
        sleep(0.1)
        print(i, 'part-time-job')
        result[i] +=' part-time-job'
print("labels10 are done!")


pr10 = work4(test, label_salary)
for i in range(len(test)):
    if pr10[i] == 1:
        sleep(0.1)
        print(i, 'salary')
        result[i] +=' salary'
print("labels11 are done!")
pr11 = work4(test, label_supervising)
for i in range(len(test)):
    if pr11[i] == 1:
        sleep(0.1)
        print(i, 'supervising-job')
        result[i] +=' supervising-job'
print("labels12 are done!")
```

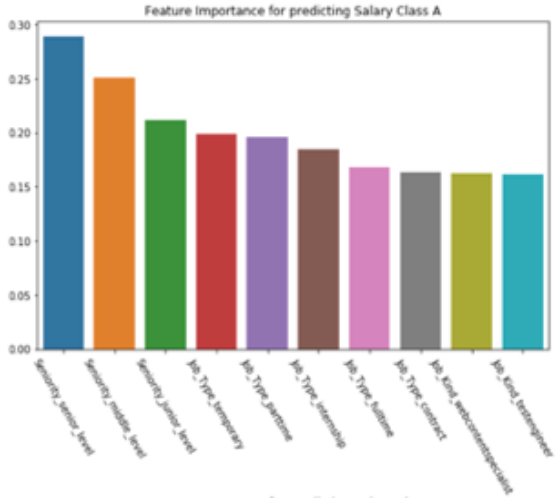
Додаток 2

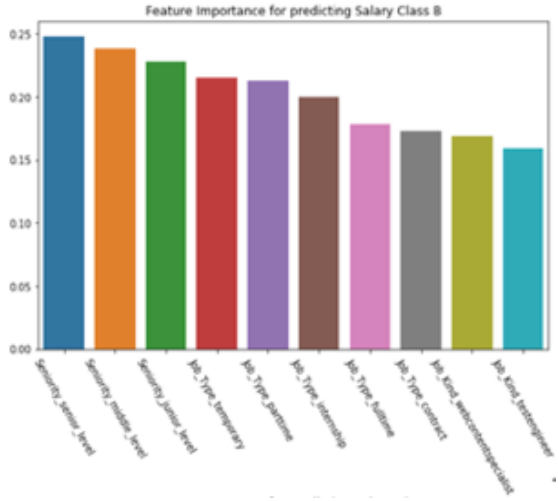
Копії графічних матеріалів

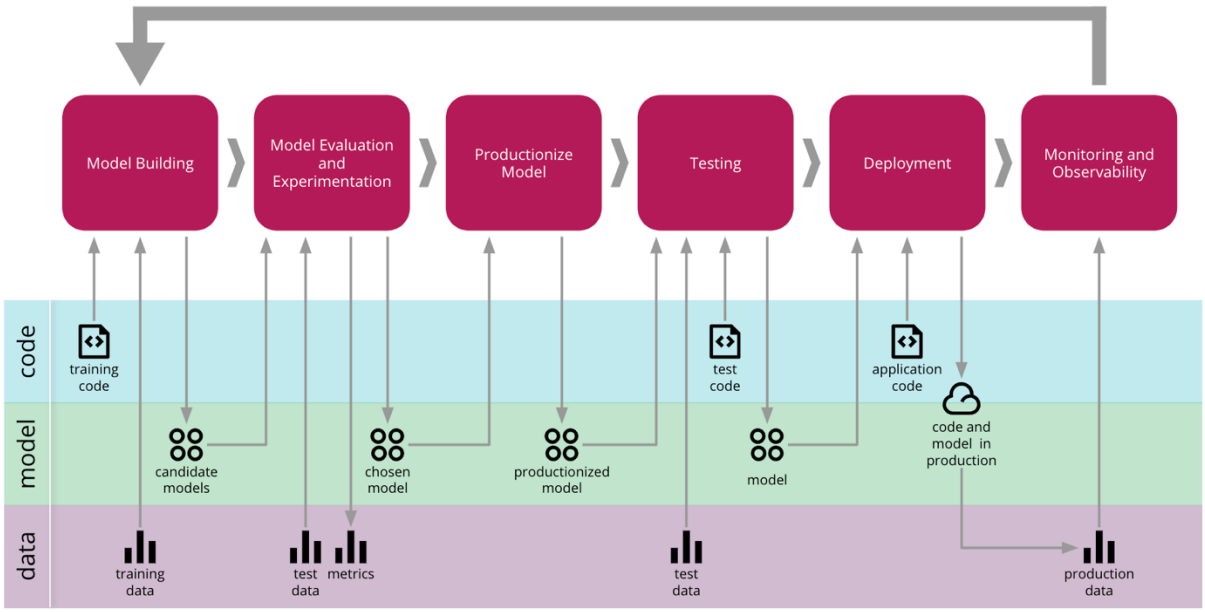

Decision Tree (depth=4)
KNN (k=7)
Kernel SVM
Soft Voting

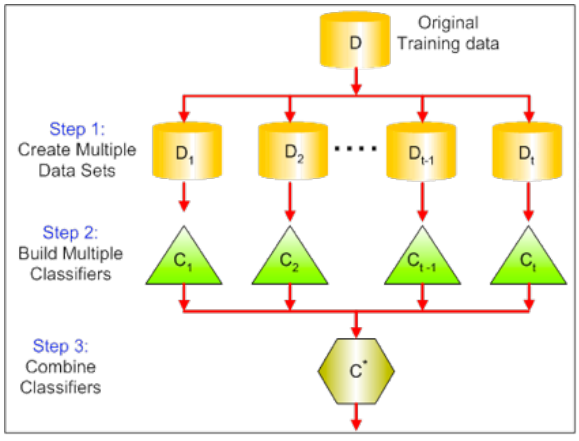

Boosting Iterations
50
100
150
RMSE (Cross-Validation)
Max Tree Depth

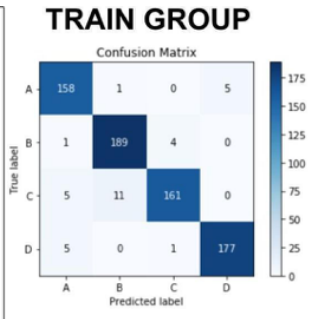

DATASET
TRAIN
MODEL
TEST
ERRORS
TRAIN
MODEL
TEST
ERRORS
TRAIN
MODEL
PREDICTION

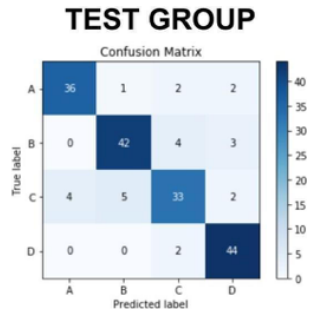

Feature Importance for predicting Salary Class A

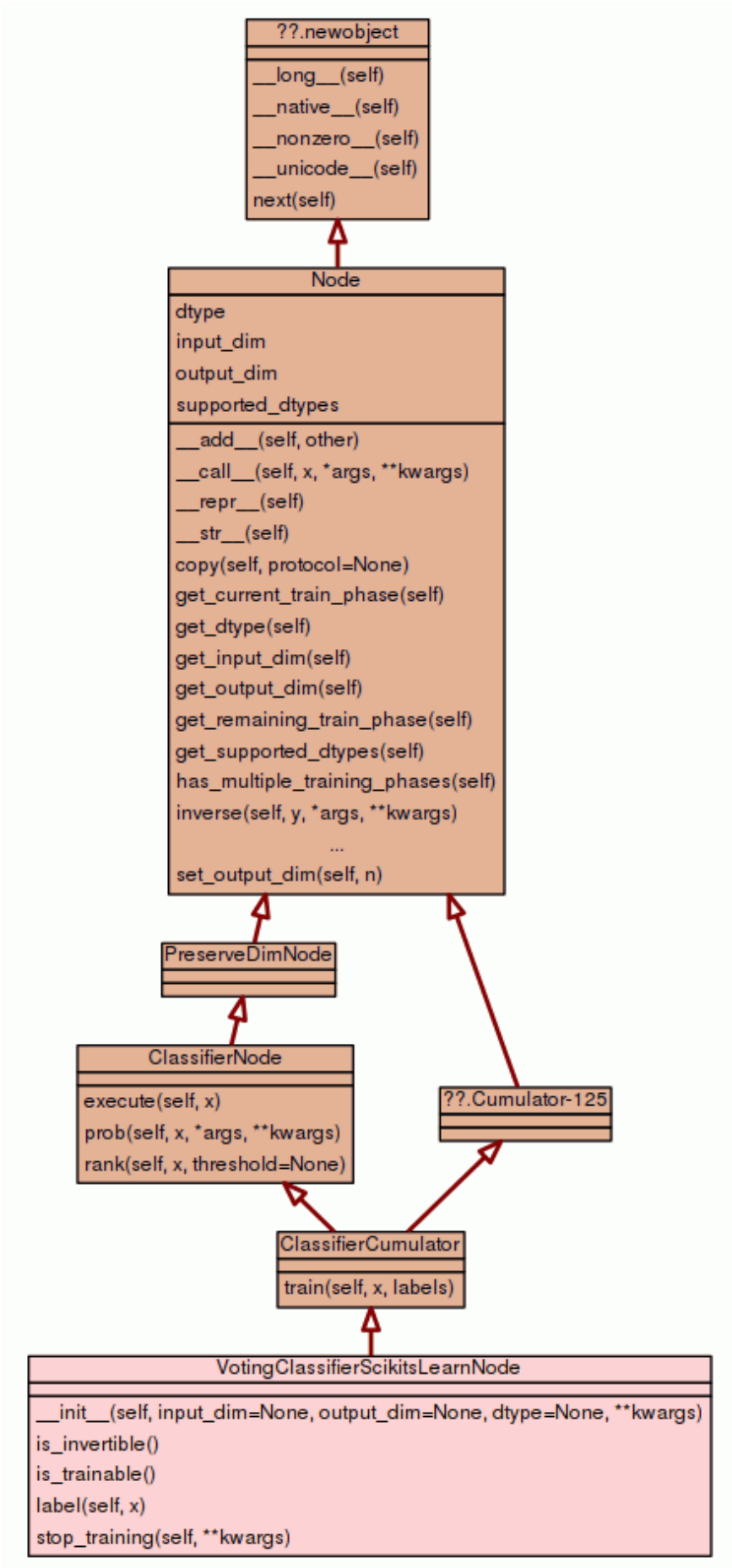

Feature Importance for predicting Salary Class B


Model Building
Model Evaluation and Experimentation
Productionize Model
Testing
Deployment
Monitoring and Observability
code
model
data
training code
candidate models
training data
test data
metrics
chosen model
productionized model
test code
test data
model
application code
code and model in production
production data


Original Training data
D
Step 1: Create Multiple Data Sets
Step 2: Build Multiple Classifiers
Step 3: Combine Classifiers
C*


TRAIN GROUP
Confusion Matrix
True label
Predicted label


TEST GROUP
Confusion Matrix
True label
Predicted label


??.newobject
__long__(self)
__native__(self)
__nonzero__(self)
__unicode__(self)
next(self)
Node
dtype
input_dim
output_dim
supported_dtypes
__add__(self, other)
__call__(self, x, *args, **kwargs)
__repr__(self)
__str__(self)
copy(self, protocol=None)
get_current_train_phase(self)
get_dtype(self)
get_input_dim(self)
get_output_dim(self)
get_remaining_train_phase(self)
get_supported_dtypes(self)
has_multiple_training_phases(self)
inverse(self, y, *args, **kwargs)
...
set_output_dim(self, n)
PreserveDimNode
ClassifierNode
execute(self, x)
prob(self, x, *args, **kwargs)
rank(self, x, threshold=None)
??.Cumulator-125
ClassifierCumulator
train(self, x, labels)
VotingClassifierScikitsLearnNode
__init__(self, input_dim=None, output_dim=None, dtype=None, **kwargs)
is_invertible()
is_trainable()
label(self, x)
stop_training(self, **kwargs)

Додаток 3

Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

СПОСІБ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ АВТОМАТИЗОВАНОГО ТЕГУВАННЯ ТЕКСТОВИХ ДОКУМЕНТІВ

Виконав: Лисогор Дмитро Юрійович

Науковий керівник: доцент, к.т.н. Заболотня Тетяна Миколаївна

Київ – 2019

Постановка науково-технічної задачі

– вивчення специфіки текстових документів;
– формулювання вимог до розроблюваної моделі для автоматизованого тегування текстових документів;
– оброблення вхідних даних та пошук додаткових параметрів в тексті для аналізу на основі геопозицій;
– розроблення моделі для аналізу та автоматизованого тегування текстових документів;
– порівняння розробленої моделі з існуючими, класичними;
– визначення вимог до відповідного програмного забезпечення, яке реалізує запропоновану модель;
– описати бізнес-модель, що дозволить представити на ринку повноцінний програмний продукт із використанням представлених у роботі напрацювань.

Об’єкт та предмет дослідження

- **Об’єктом дослідження** є процес використання тексту для аналізу та автоматизованого тегування даних.

- **Предметом дослідження** є методи та підходи до класифікації та тегування текстових документів.

Мета дослідження

- Вдосконалення процесу пошуку категорій у тексті для отримання точних результатів шляхом розроблення та впровадження нового методу для автоматизованого виділення тегів.

Дослідження

- Information Retrieval, Cambridge University Press.(2008), Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze
- Detecting Text with Connectionist Text Proposal Network(2016), Zhi Tian , Weilin Huang , Tong He , Pan H, and Yu Q
- Long short-term memory recurrent neural network architectures for large scale acoustic modeling(2014), Hasim Sak et al., Google Research
- A survey of named entity recognition and classification(2012), David Nadeau, Satoshi Sekine

Існуючі рішення проблеми

INDEED

DOU

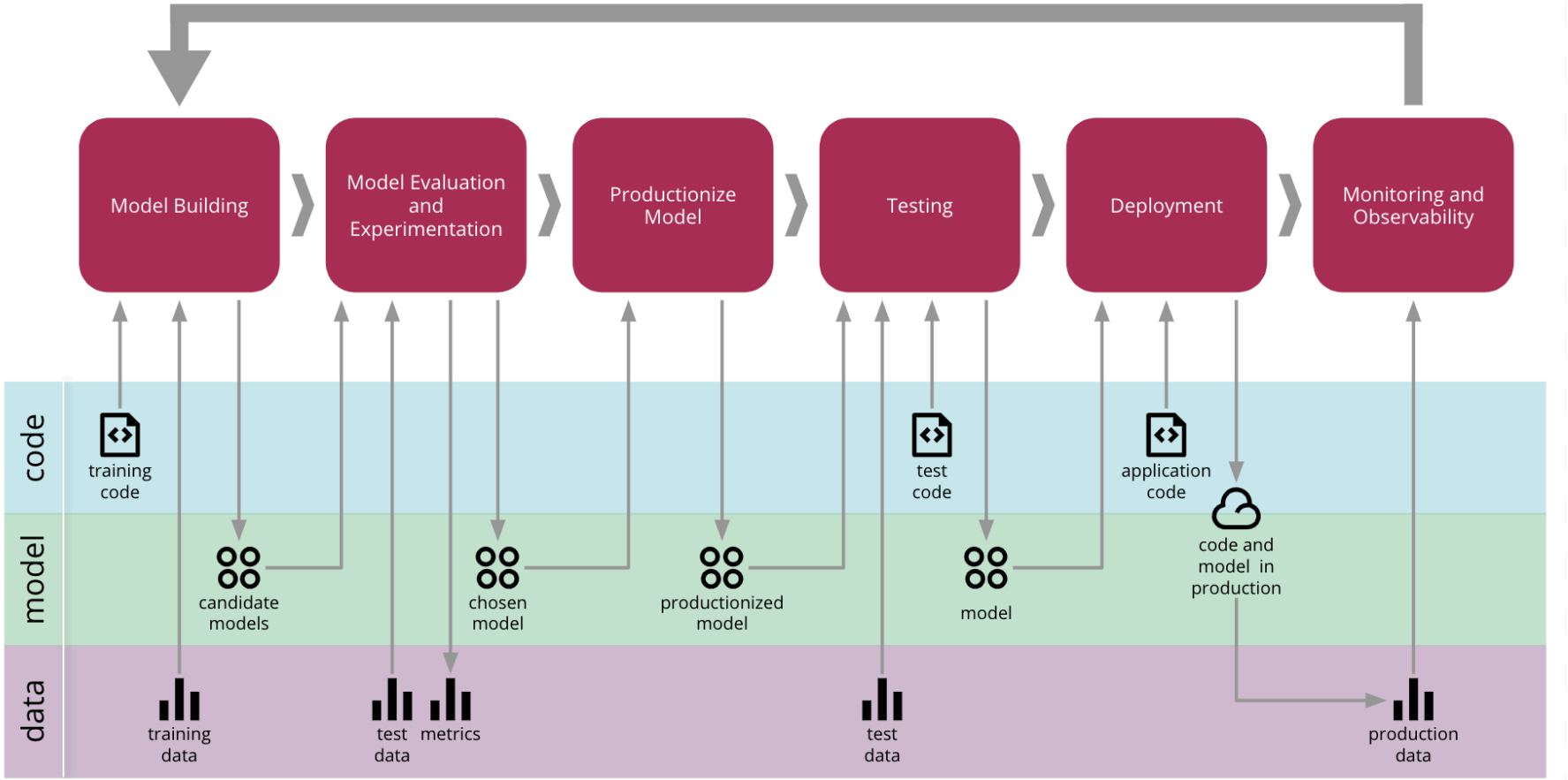
Термінологія

- KNN model with bagging (Named entity recognition) – метрична модель для автоматизованої класифікації об'єктів або регресії. У разі використання моделі для класифікації, об'єкт присвоюється до того класу, який є найбільш поширеним серед k-сусідів, класи яких вже відомі або попередньо визначені.
- Модель дерева рішень — модель, в якій кожне рішення базується на порівнянні двох чисел за постійний час. Її використовують для встановлення обчислювальної складності сортування та пошуку

Запропонований спосіб до вирішення поставленої задачі

- Формулювання вимог до розроблюваного застосунку для автоматизованого тегування текстових документів
- Підготовка та очищення даних для моделі машинного навчання
- Аналіз даних та вибір моделей для класифікації текстових документів.
- На основі вказаних категорій та опису генерувати відповідний список текстових документів.

Етапи розробки ПЗ для автоматизованої категоризації текстових документів



Критерій ефективності

- В якості критерія ефективності в даній роботі використовуватимемо міру F1, яка у статистичному аналізі двійкової класифікації є мірою точності визначення категорій, які присутні в тексті .

Підготовка та очищення даних для навчання моделі

- Перед аналізом ефективності вищеназваних моделей для автоматизованого тегування текстових документів, необхідно підготувати дані для навчання моделі та провести аналіз цих даних

Підготовка та очищення даних для навчання моделі

- Базовий набір – містить основну інформацію, очищену від усіх загальних назв та деяких інших полів, які знаходяться в публікації чи документі;

Підготовка та очищення даних для навчання моделі

- Векторизований набір – містить кожне слово, яке з’явилося в будь-якій публікації чи документі, перетворене на стовпець в таблиці даних, де значення відповідної категорії еквівалентне відповідно до кількості разів, скільки кожне слово з’явилося загалом в тексті;

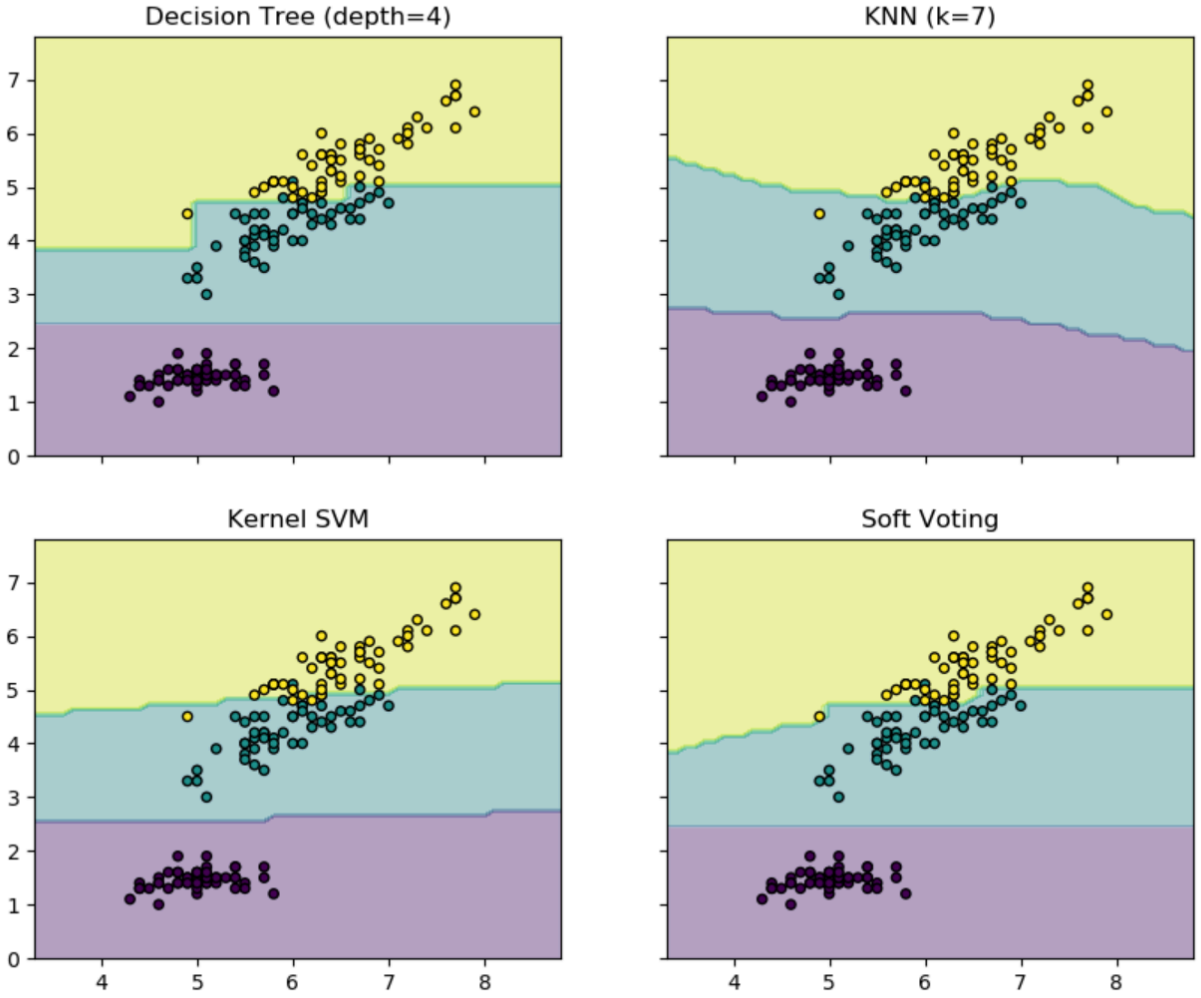
Підготовка та очищення даних для навчання моделі

- З базового і векторизованого наборів даних надалі буде проведений аналіз та вибір моделей для реалізації застосунку для автоматизованого тегування текстових документів.

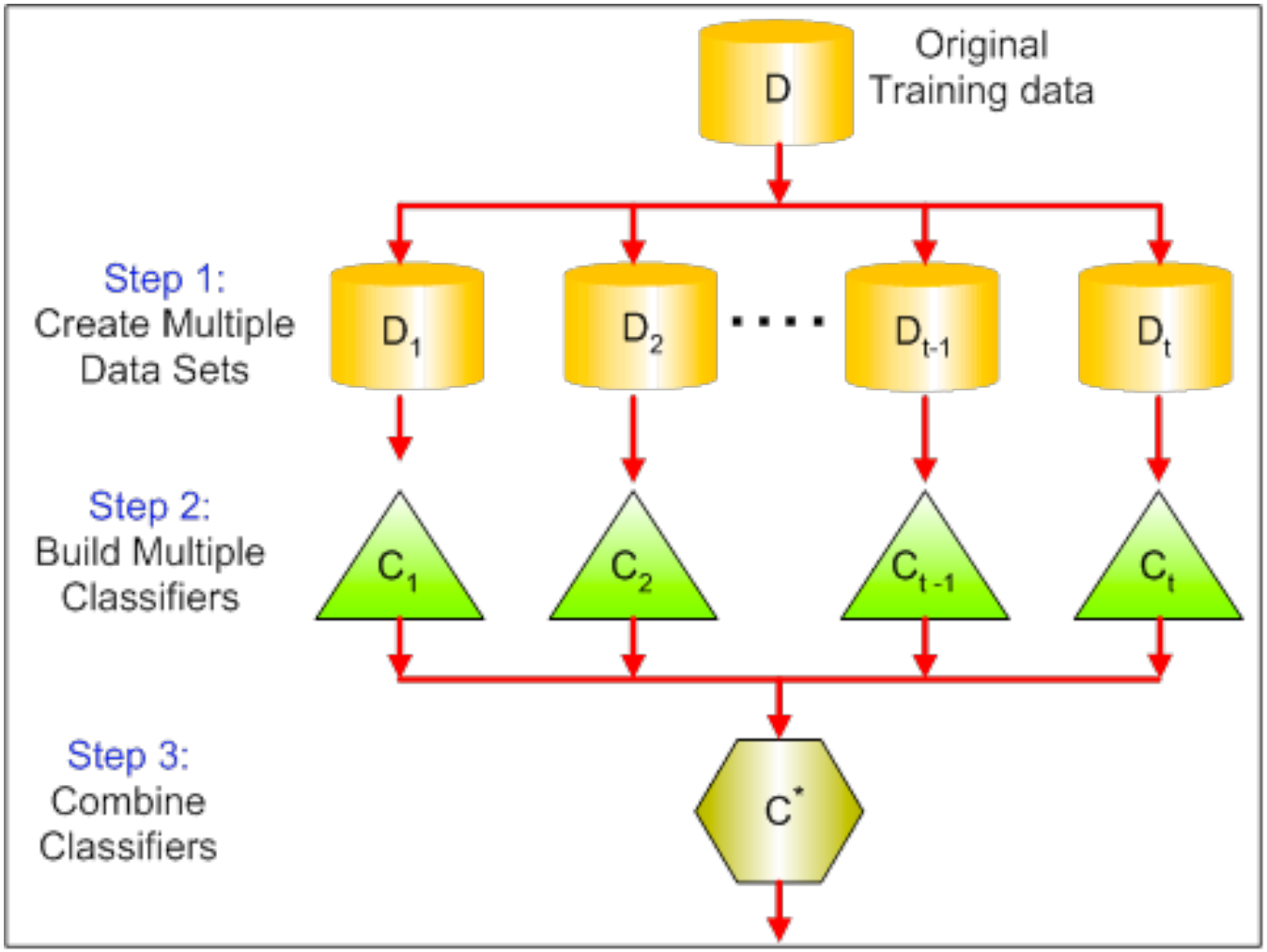
Побудова класифікатору голосування

Classifier	Class 1	Class 2	Class 3
Classifier 1	w1 * 0.2	w1 * 0.5	w1 * 0.3
Classifier 2	w2 * 0.2	w2 * 0.2	w2 * 0.2
Classifier 3	w3 * 0.3	w3 * 0.4	w3 * 0.3
Class 1	0.37	0.4	0.23

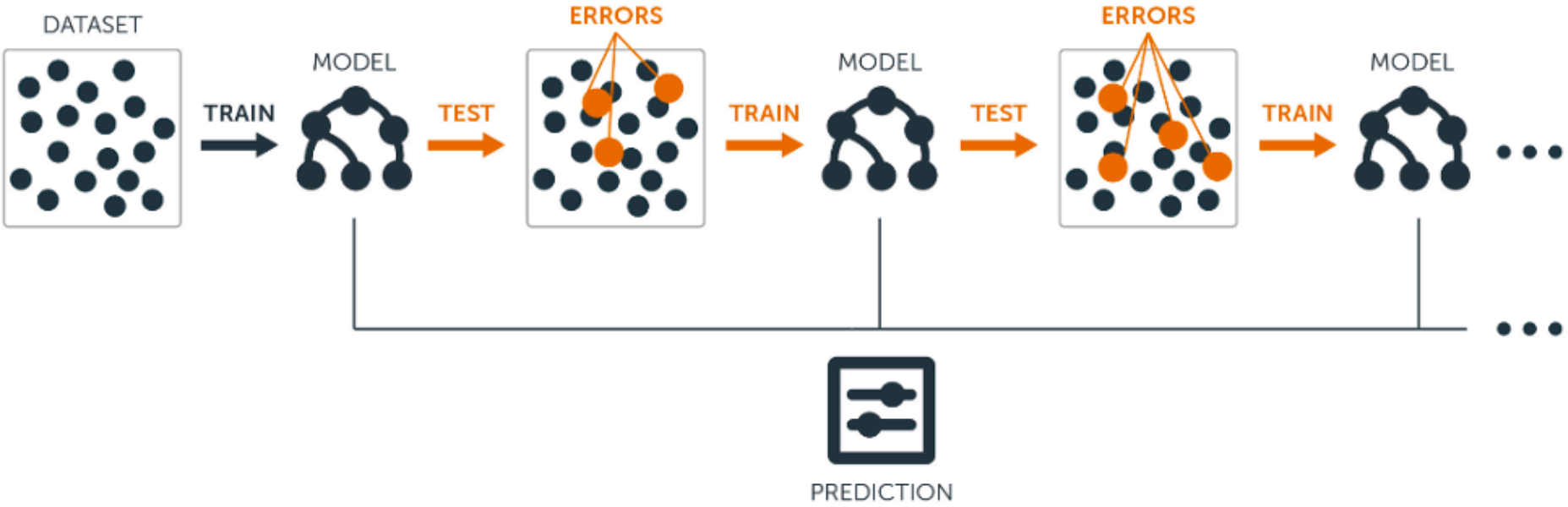
Побудова класифікатору голосування



Архітектура ансамблю класифікаторів



Процедура навчання комбінованої моделі



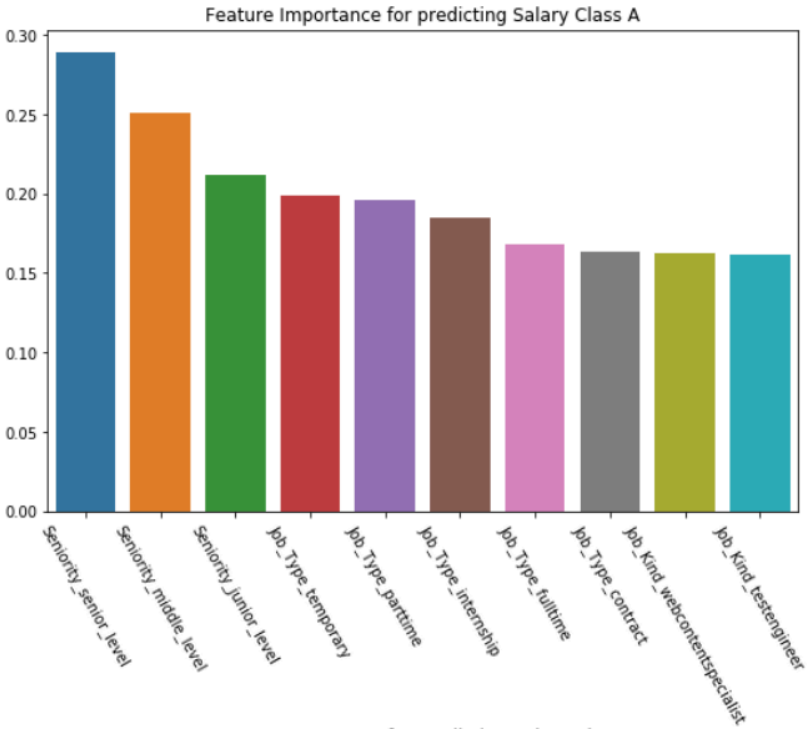
АНАЛІЗ РЕАЛІЗАЦІЇ МЕТОДУ ТА ОТРИМАНИХ РЕЗУЛЬТАТІВ

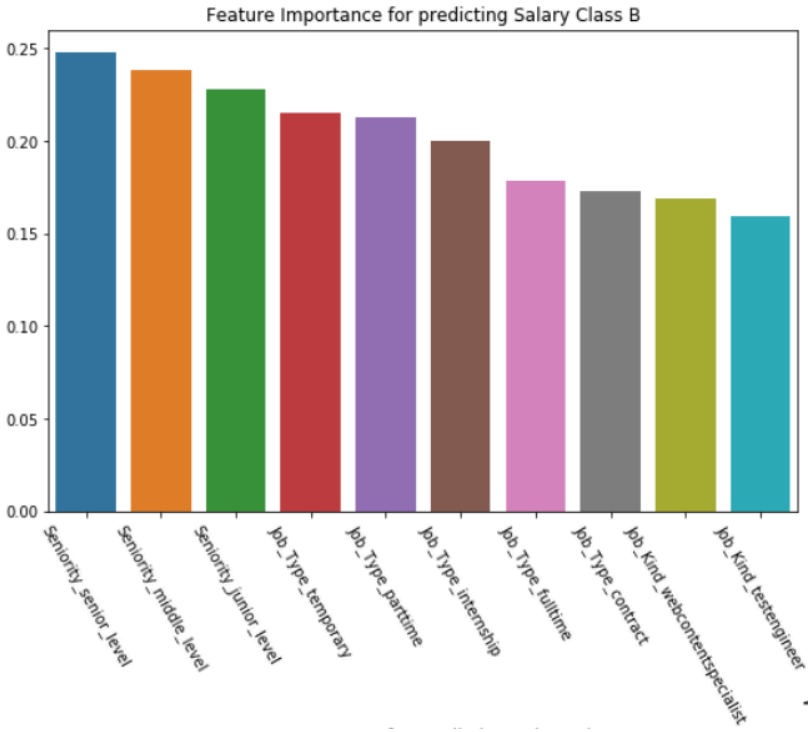
- Для перевірки доцільності роботи ансамблю моделей для автоматизованого тегування текстових даних було обрано сайт indeed.com для збору даних. Сайт пропонує найбільшу в світі пошукову систему. Теги знайдені на сайті дозволяють користувачам фільтрувати та легше знаходити роботу

Аналіз текстових документів та вибір моделей

- На початковому етапі аналізу даних використовуємо всі моделі зі списку визначеному вище для роботи з даними, але обрано одну як базову - логістичну регресію з перехресною валідацією

Визначення важливості міток серед векторизованого набору даних





Діаграма визначення важливості міток серед векторизованого набору даних

Важливі властивості визначені з діаграми

- Стаж – як ми бачимо, досвід роботи впливає дуже сильно на всі категорії, будучи першим за важливістю коефіцієнтом за абсолютною величиною;
- Тип роботи (Job_Type);
- Локація чи координати компанії;

Комбінація моделей та визначення критерію ефективності

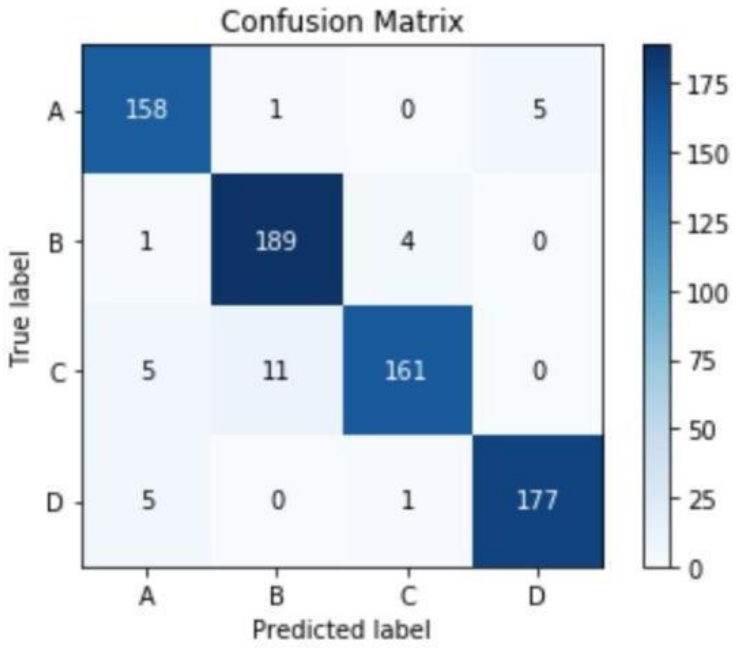
- a KNN model with bagging;
- a Decision Tree model with boosting.

Комбінація моделей та визначення критерію ефективності

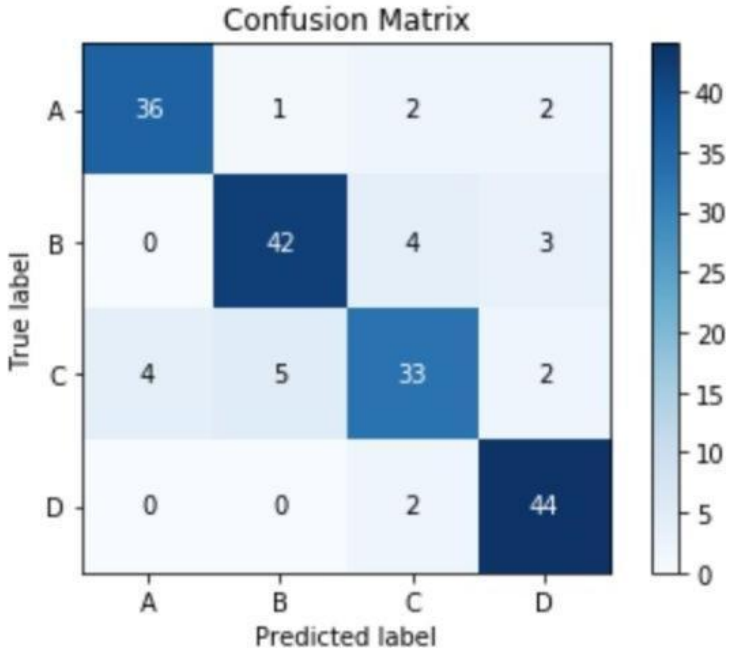
Середня точність знаходження основних категорій вакансій, визначених комбінацією вищезгаданих моделей (або ансамблю моделей) є 86%, що означає, що принаймні більше 80% усіх прогнозованих значень для кожної категорії було правильним.

Матриця відхилень спрогнозованих категорій

TRAIN GROUP



TEST GROUP



Науково-іноваційна новизна

1.Розроблено спосіб автоматизованої категоризації текстових документів, що відрізняється від існуючих комбінацією існуючих класичних моделей класифікації текстових даних, які показують точніший результат виконання.

2. Запропоновано ансамбль моделей для автоматизованої категоризації даних, який відрізняється від відомих меншою обчислювальною складністю та точнішим результатом роботи з даними, які залежать від часу.

Висновки

- Розглянуто задачу розроблення програмного забезпечення для автоматизованого тегування текстових документів, виділено існуючі базові моделі для категоризації текстів та запропоновані кроки щодо їх адаптації для пошуку категорій у текстових документах
- Сформовано кроки для вирішення поставленої задачі
- Виконано реалізацію та проведено вивчення ефективності комбінацій моделей обробки текстових документів для підвищення точності знаходження категорій

Апробування отриманих результатів

- XII наукова конференція магістрантів та аспірантів «Прикладна математика та комп’ютинг» (ПМК-2019).

ДЯКУЮ ЗА УВАГУ!